Modelling and optimisation of the design and topology of flexible frames with rigid joints

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)
im Fach Mathematik

eingereicht an der
Mathematisch-Wissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herrn Kshitij Kulshreshtha
geboren am 26. September 1979 in Aligarh, Indien

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h. c. Christoph Markschies

Dekan der Mathematisch-Wissenschaftlichen Fakultät II:
Prof. Dr. Peter Frensch

Gutachter:
1. Prof. Dr. Andreas Griewank (Humboldt-Universität zu Berlin)
2. Prof. Dr. Nicolas R. Gauger (Humboldt-Universität zu Berlin)
3. Prof. Dr. Mohamed Masmoudi (Université de Toulouse)

eingereicht am: 23. Februar 2010
Tag der mündlichen Prüfung: 8. Oktober 2010

Dedicated to my parents; and to friends,
who helped me keep my humour.

Gewidmet an meine Eltern; und an Freunde,
die mir halfen, mein Humor zu bewahren.

मेरे माता–पिता को, और उन मित्रों को समर्पित,
जिन्होंने मुझे विनोदी बनाए रखा।

Acknowledgements

I would like to express my deepest thanks all the present and past members of Prof. A. Griewank's research group at the Humboldt Univesity, most of all himself, his post-doctoral colleague Dr. S. Körkel and also his students, as well as all the members and students of the DFG Research Training Group "Analysis, Numerics and Optimization of Multiphase Problems", whose help and cooperation was indispensable while I conducted my research. A special thanks also goes to J. Riehme, who allowed me to use his original although experimental AD-tool.

Abstract

Structural optimisation currently relies heavily on methods based on discretisation. In simpler cases like the simulation of frames and trusses, where discretisation is not necessary, only the elongation or compression is considered and the joints are free, like ball and socket joints, in order to avoid bending the trusses. In this dissertation a discretisation free method for the modelling and optimisation of frames is developed which considers bending of the beams along with compression or elongation with joints between the beams being rigid. Rigid joints are commonly the result of welding two beams together or connecting them using mutiple rivets. The optimisation problems, both state and design optimisation, are formulated via the total elastic energy and the work done by external forces. Moreover, for the optimal sizing problem a topological sensitivity for introduction of new beams between any two arbitrary positions in the frame is discussed.

Zusammenfassung

Strukturoptimierung ist momentan stark auf Diskretisierungsmethoden angewiesen. In einfachen Fällen, wie die Simulation von Rahmen und Stabwerke, wo eine Diskretisierung nicht notwendig ist, werden nur die Dehnung oder die Stauchung der Stäbe betrachtet, und die Verbindungen sind frei, wie Kugelgelenke, um die Biegungen der Stäbe zu vermeiden. In dieser Dissertation wird eine diskretisierungsfreie Methode zur Modellierung und Optimierung eines Rahmens entwickelt, die die Biegung der Balken sowie die Dehnung oder Stauchung zusammen betrachtet, wobei starre Verbindungen angenommen werden. Starre Verbindungen entstehen, wenn die Balken zusammen geschweißt oder mit mehrere Nieten verbunden sind. Die Optimierungsprobleme, sowohl das Zustands- und als auch das Entwurfsproblem, sind durch die gesamte elastische Energie und die Arbeit der äußeren Kräfte gegeben. Für das Problem der optimalen Größeneinteilung wird darüber hinaus eine topologische Sensitivität zur Einführung neuer Balken zwischen zwei beliebigen Punkten auf dem Rahmen diskutiert.

Contents

List of Figures

List of Tables

Preface

At the end of my Master Thesis work I was inclined to continue working on solution methods for problems related to mechanics. Thus upon arrival in Prof. Griewank's Institute for Scientific Computing in Dresden, I looked for a project in that area. However, after looking up a large number of publications and talking to engineers on problems like external acoustics of shells, I finally came to the conclusion that I should in fact start in a completely new area, the area of structural and topological optimisation. Incidentally at the same time Prof. Griewank moved to Berlin and happened to meet Dr. Klaus Schröer at a workshop. Dr. Schröer, who is in the production department of the Volkswagen AG, was interested in developing a rather rudimentary model based on beams for the structure of a car frame, and do structural optimisation on this model, in terms of saving cost of material and production while attaining the same physical properties of strength and compliance. I took up this idea in early 2005 as my Ph.D. project.

The literature survey on the subject of modelling beams and frames lasted quite a bit, but did not result in a lot of useful information. As almost all the literature was about modelling structures using finite element models, which have their merits, and took rather long to solve for, it made them unsuitable for interactive optimisation. By the end of 2005 we had come to the conclusion that we should develop our own nonlinear model for the energy stored in a deformed beam, given the state of the beam in terms of its endpoint positions and orientations, before we can use this model for computing topological sensitivities to the end of structural optimisation.

The project was funded by Volkswagen for a period of six months starting January 2006, and later by the resources of Prof. Griewank's Max Plank prize, DFG Research Centre Matheon, and the DFG Research Training Group 1128 on Multiphase Problems.

Chapter 1

A short overview of structural optimisation

1.1 Beam based models

Topology design of beam based models like trus structures has been discussed in detail in BENDSØE & SIGMUND [2003]. Trusses are in the form of grid like continua and their study dates back to the beginning of the last century [MICHELL, 1904]. Numerical methods for discrete truss topology problems were devised in DORN et al. [1964] and FLERON [1964] and developed into a well established theory for frames [ACHTZIGER, 1993, 1996].

The optimisation of the geometry and topology of trusses is formulated as the ground structure method that allows for a layout of trusses as a certain set of connections between a fixed set of nodal points or by vanishing them. The geometry allows for using a continuously varying cross-sectional bar area with the possibility of a zero area thus forming a standard sizing problem. The truss topology problems were formulated in terms of member forces arising from compression or elongation of the bars, ignoring kinematic compatibility to obtain linear programming problems in member areas and forces. In a displacement based formulation small non zero lower bounds on areas have been imposed in order to have a positive definite stiffness matrix. However, such a topology design problem is unusual as a structural optimisation problem as the number of design variables is typically several magnitudes bigger than the number of state variables describing the equilibrium of the structure. Also, the stiffness matrix of the ground structure with certain members at zero gauge can be singular and will be singular for most optimal designs when viewed as a part of the ground structure. BENDSØE & SIGMUND [2003, Chapter 4] gives a detailed description of the problem formulation.

1.2 Finite element based models

Alternatively structural optimisation problems may be formulated as material distribution problems. The key is a continuous material distribution, for which an optimality criterion is developed. It is typically later discretised as finite elements. The geometry is represented by the black and white raster of *voxels* given by the finite element discretisation. Most commonly the density of the material is used as the design variable in each voxel. The number of design variable is comparable to that of the state variables, both of which are related to the total number of voxels used in the domain. In large 3D models these can grow astonishingly fast, especially if adaptive refinement strategies are used in order to reduce error. Post processing of the optimal design is also needed in order to avoid checkerboarding due to mesh dependence. BENDSØE & SIGMUND [2003, Chapter 1, 2, 3] discusses this approach in detail and extends the method to anisotropic material distribution problems using the concept of homogenisation.

1.3 Introduction to our approach

As we see above numerical methods for design optimisation are frequently computed and verified on frameworks under purely extensional or compressional forces. This kind of modelling is unsuitable for flexible frames as no bending effects are taken into account during the computation. On the other hand going to a full 3-D Finite Element modelling, which is used extensively for modelling large structures like car bodies, depends on such a huge number of design and state variables that an interactive optimisation process is completely impossible. We are thus pursuing the goal of modelling a car frame using a framework of flexible beams joint together with either flexible or rigid joints at their ends. Although the classical linear elastic theory for beams is well known, [ANTMAN, 2005; BAŽANT & CEDOLIN, 1991; TIMOSHENKO, 1928] its conversion to an efficient computational model turns out to be unexpectedly complicated. Particularly an internal balance between the bending and change in length must be achieved, which is simply ignored in a first order model. However such a relationship between the bending behaviour and the change in length behaviour can be controlled and simulated in the design process using various hollow cross-sectional profiles or use of different kinds of steels.

Let us consider the effects on a rectangular frame made of four beams with

right-angular joints at the corners. If two opposing beams are stronger than the other two and are pushed towards each other keeping them parallel, then the crosswise thinner beams shall first be compressed upto a critical limit, after which they shall take an S-form to achieve balance between bending and compression. If the stronger beams are constrained against sideways movement then the weaker ones shall form a double-S form, which will then take an omega form if the force continues to grow. Our goal is to model these effects realistically without actually discretising the beam into piecewise elements and approximating.

This dissertation is organised as follows. chapter 2 shortly recapitulates the ideas involved in algorithmic differentiation as well as introduces the notion of *nested differentiation*. In chapter 3 the total quasi-Newton method for solving nonlinear constrained optimisation problem has been briefly described. A software implementation thereof, `LRAMBO`, was used to implement the solution of the optimisation problems developed in the following chapters. A detailed model for simulating the compression, elongation and bending of a single beam in its own internal frame of reference is presented in chapter 4. This is followed by a description of how to put single beams together along with external forces and rigid joints in chapter 5. The state and design optimisation problems are formulated in chapter 6 along with a discussion of the topological sensitivity for adding new beams in the frame. In chapter 7 the exact expressions for the derivatives, which may be used instead of using algorithmic differentiation, have been derived. chapter 8 finally presents some observations from the numerical implementation of the modelling and optimisation, concluding in chapter 9 with a few remarks about the model, its optimisation, some observations and directions for future research.

Chapter 2

Computation of derivatives

2.1 Basics of algorithmic differentiation

For the purpose of evaluating the derivative required for the Newton iteration and the derivatives of the Elastic energy needed later, we use algorithmic differentiation (AD). For any function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, which can be computed via an algorithm or computer program code, AD allows the efficient evaluation of directional derivatives (first or higher order) at a fixed evaluation point $x = (x_i)_{i=1}^n$ as products $J(x) \cdot \dot{x}$ or $\bar{y} \cdot J(x)^\top$, where the Jacobian J contains all first order partial derivatives

$$\left(\frac{\mathrm{d}f_j(x)}{\mathrm{d}x_n}\right)_{\substack{i=1\ldots n \\ j=1\ldots m}} = f'(x) = J(x) \in \mathbb{R}^{m\times n}$$

of the function f at the evaluation point x.

For that the code representing f is decomposed into a code list of atomic operations

$$v_j = \varphi_j(v_i)_{i\prec j}, \text{ for } j = 1, \ldots, q.$$

The elemental functions φ represents the arithmetic operators ($+$ or $*$), and intrinsic standard functions (`sin`, `cos` etc.). Every single variable v_j will hold the result of exactly one elemental function φ_j applied to their arguments $\{v_i | i \prec j\}$, where the relation $i \prec j$ can be read as *v_i is directly used to compute v_j* or *v_i has direct impact on v_j*. Variables $v_i = x_i$ with indices $i = 1-n, \ldots, 0$, represents the independent variables from $x \in \mathbb{R}^n$, whereas variables computed in the last m steps of the code list represents the dependents of the result vector

v_{-1}	$=$	x_1	$\dot{v}_{-1}$	$=$	$\dot{x}_1$
v_0	$=$	x_2	$\dot{v}_0$	$=$	$\dot{x}_2$
v_1	$=$	$v_{-1} * v_0$	$\dot{v}_1$	$=$	$v_{-1} * \dot{v}_0 + \dot{v}_{-1} * v_0$
v_2	$=$	$\sin(v_1)$	$\dot{v}_2$	$=$	$\cos(v_1) * \dot{v}_1$
v_3	$=$	$v_2 * v_0$	$\dot{v}_3$	$=$	$\dot{v}_2 * v_0 + \dot{v}_2 * v_0$
(a)			(b)		

Table 2.1: Code list and augmentation for function (2.1)

$y = f(x) \in \mathbb{R}^m$. Table 2.1a shows the code list for the function

$$f : \mathbb{R}^2 \mapsto \mathbb{R}^1 \quad : \quad f(x_1, x_2) = sin(x_1 * x_2) * x_2 \tag{2.1}$$

In order to compute derivatives together with function values the code list is augmented with additional statements that carry derivatives through the computation. There are mainly two different augmentation modes for AD: In the *Forward* or *Tangent Linear Mode* of AD initial derivatives $\dot{x}_i$ for the function arguments will be propagated by combining the evaluation of elemental functions φ_j with the computation of the derivative

$$\dot{v}_j = \sum_{i \prec j} \frac{\mathrm{d}\varphi_j}{\mathrm{d}v_i} * \dot{v}_i \tag{2.2}$$

according to the chain rule, whereas the *local partial derivatives* $\mathrm{d}\varphi_j / \mathrm{d}v_i$ of elemental functions φ_j corresponds with the differentiation rules of basic calculus. Table 2.1b shows the augmentations required for the code list in Table 2.1a in order to compute the product of the Jacobian $J(x)$ of f with the initial tangent $\dot{x}$ for (2.1) at evaluation point x.

The so called *Reverse* or *Adjoint Mode* of AD allows to compute the product $\bar{y} \cdot J(x)^\top$ of an initial adjoint $\bar{y}$ with the transposed Jacobian $J^\top$ of f at a small multiple of the computational cost of f itself. But the generation of adjoint code is usually much more complicated than forward mode, since the adjoints $\bar{v}_j$ of the intermediate variables have to be computed in reverse order to propagate the initial adjoint $\bar{y}$ of the function result $y = f(x)$ to the adjoints $\bar{x}$ of the inputs x. This raises a couple of technical problems, for instance all values overwritten during the forward sweep need to be stored.

AD-tools fall commonly into one of two classes: tools based on *operator overloading* and *source code transformers*. The latter takes the original source code

provided for f and generate a new source (often of the same higher level programming language as the original one) that evaluates also derivatives of f. Famous examples are `ADIFOR` [BISCHOF et al., 1992, 1994], `Tapenade` [HASCOET, 2004]), `TAF` [GIERING, 1997; GIERING & KAMINSKI, 1998] for Fortran codes, `TAMC` (related to `TAF` [GIERING, 1997]) and `ADIC` [BISCHOF & ROH, 1997] for `C` and `C++` codes. AD-tools based on operator overloading require a programming language that supports the creation of user defined data types together with the definition of operators for user defined types. This allows to define new meanings for the atomic operations if they are applied to operands of the user defined data type. Since the new meaning does not conflict with the original meaning of intrinsic operators applied on intrinsic types, this technique is called *operator overloading*. Code generated by the compiler to evaluate arithmetic expressions is a code list in fact, thus a simple implementation of the forward mode of AD might augment the code list by combining the evaluation of φ_j and $\dot{\varphi}_j$ in the overloading operators. Well known tools based on overloading are `ADOL-C` [GRIEWANK et al., 1996], `CppAD` [BELL, 2003–2010], and `FADBAD` [STAUNING, 1997] for `C++`, `AD01` for Fortran codes. See GRIEWANK & WALTHER [2008] for a comprehensive description of AD, and `http://www.autodiff.org` for information about current developments and existing tools for AD.

2.2 Nested algorithmic differentiation

For our purpose however we use a new tool `ADTAGEO`, originally developed by J. Riehme and A. Griewank. This new tool is used because of the special needs of the model, where we require *nested derivatives*. Derivatives are *nested*, e.g. in a situation when the function $y = f(x)$, whose derivatives are to be computed by AD, can only be evaluated as the root of some other implicit parametrised function $G_x(y)$. Here, the evaluation of the root of the parametric function $G_x(y)$ for a given $\bar{x}$ using the Newton's method requires the inner derivative $\dfrac{\mathrm{d}G_{\bar{x}}}{\mathrm{d}y}(y)$ in each iteration

$$y_+ = y - \left(\frac{\mathrm{d}G_{\bar{x}}}{\mathrm{d}y}(y)\right)^{-1} G_{\bar{x}}(y)$$

Then at the solution point $\bar{y} = f(\bar{x})$ we compute the outer derivative $\dfrac{\mathrm{d}f}{\mathrm{d}x}(\bar{x})$. For the optimization algorithm we shall need the derivatives of the elastic energy (see the following sections) with respect to the state and design variables.

However to evaluate the elastic energy for a given state we need to solve a non linear equation, where we require internal derivatives for the Newton iterations. Such nested derivatives are impossible to compute with any of the AD-Tools mentioned in the previous section.

ADTAGEO utilises operator overloading in C++ to build a representation of the code list known as *computational graph*. Every intermediate variable $v_j = \varphi(v_i)_{i \prec j}$, $j = 1-n, \ldots, q$ represents a vertex v_j, and the dependency relation $i \prec j$ introduces directed arcs (i, j) between the arguments of the elemental function φ_j and their result v_j. Figure 2.1a shows the computational graph of the code list from 2.1a. If all arcs (i, j) are annotated with the values of the corresponding *local partial derivatives* $\frac{\mathrm{d}}{\mathrm{d}v_i}\varphi_j(v_i)_{i \prec j}$, we get a linearised computational graph of the Jacobian $J(x)$ of function F at the evaluation point x (Figure 2.1). Derivatives can be evaluated by propagating initial tangents $\dot{x}$ from vertices v_i, $i = 1-n, \ldots, 0$, associated with the independent variables (forward mode), or by propagating adjoints from the dependent vertices v_i, $i = q-m, \ldots, q$ backwards to the independents (reverse mode propagation).

Tools based on overloading differ mostly in their technique to store the computational graph, which is in fact a directed acyclic graph (*DAG*) with paths from the independent variables x towards the dependents $y = f(x)$ formed by sequences of arcs (e.g. directed edges) between intermediate variables. Depending on the concrete implementation various propagation techniques might be required. So far all overloading AD-tools have in common a static behaviour of storing the complete computational graph first, followed by at least one accumulation sweep at the end of the evaluation of f to build the desired derivatives.

In contrast ADTAGEO is based strictly on *Instant Elimination*: Whenever a program variable is overwritten or deallocated, the corresponding vertex will be eliminated from the DAG instantaneously according to the *Vertex Elimination Rule* [GRIEWANK & WALTHER, 2008]. Figure 2.1 illustrates the elimination of v_2 followed by v_1. The computational graph grows with any atomic operation by one vertex (and at most 2 arcs). Removing vertices of dying variables creates a DAG, that contains only vertices associated with variables alive at any point in time of the program execution. We call such a graph a *Life-DAG*, and the removing of vertices at the end of the lifespan of their variable is called *Instant Elimination*.

If variables in the program are allocated and deallocated in a clever way, the Life-DAG might become bipartite, i.e. every path in the DAG is a single arc

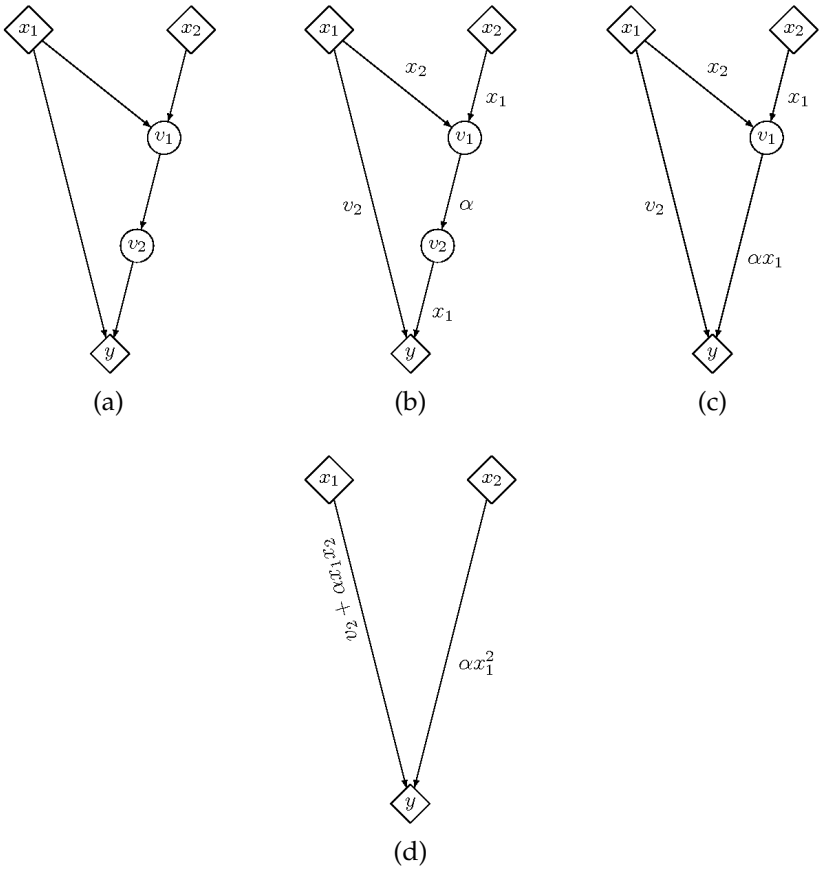


Figure 2.1: Vertex elimination: (a) Computational graph of code list from 2.1a. (b) DAG augmented with local partials, $\alpha = cos(v_2)$. (c) DAG after elimination of vertex v_2. (d) Bipartite DAG after elimination of vertex v_1.

only (see Figure 2.1d). Than the local partial stored at the arc (i, j) is the fully accumulated partial derivative of v_j with respect to v_i and can be extracted from the Life-DAG directly.

If the Life-DAG is not bipartite, there exist paths containing at least two arcs. Thus some of the arc attributes are not fully accumulated derivatives, since their value is only a portion of the sum in (2.2). Moreover vertices not connected by an arc directly might be connected by paths, and dependencies exist even if no arc connects them directly. For that situation `ADTAGEO` implements forward and reverse propagation techniques to compute directional derivatives efficiently on non bipartite Life-DAGs too.

Moreover, whenever two vertices v_i and v_j are not connected by a least one path, than there is no dependency of v_j an v_i, and the partial derivative vanishes. Thus the sparsity structure of the Jacobian is preserved in the Life-DAG,

and will be exploited by any propagation sweep automatically. The programmer has explicit control of the propagation mode, he can choose which variables he wants to deallocate and eliminate from the Life-DAG and in what order. Nodes that upon elimination will cause a fill-in in the Jacobian can be explicitly kept alive by clever programming. Such cross country vertex elimination may save some arithmetic operations required to accumulate the jacobian as compared to forward or reverse [GRIEWANK & NAUMANN, 2002; NAUMANN, 2008; NAUMANN & HU, 2008].

As an immediate consequence of Instant Elimination the distinction between variables of the program and vertices in the computational graph is no longer necessary, since every variable has exactly one corresponding vertex v_j in the Life-DAG. (Though this might be not connected with others if the variable is allocated but has not got a value so far.) Moreover, all vertices in a Life-DAG have a unique corresponding declared and allocated variable, even if the variable is inaccessible due to scope and visibility issues.

This one-to-one association between vertices v_j of the Life-DAG and variables (for instance x_1, x_2, and y) declared in the user program makes *Nested Differentiation* possible in `ADTAGEO`. This is a unique feature: at any point during the execution derivatives can be calculated by traversals within the Life-DAG, which is kept unchanged during these propagations. The Newton iteration, where the derivative $\frac{\mathrm{d}\widetilde{\mathcal{F}}}{\mathrm{d}W}(W)$ are needed in order to compute the root W of $\mathcal{F}(W)$, is embedded in an energy computation that depends on this root. The energy itself is used in an iterative minimisation process that needs other derivatives (see section 4.6, section 5.4 and chapter 7).

Without instant vertex elimination the start and end vertices, which need to be traversed in the graph in order to propagate an internal derivative, may be temporaries that may not be associated to any declared variable at all, or their association to a variable may be ambiguous during the computation process of f. A traversal can be made only at the end of the computation of f when the complete DAG has been stored with pre-marked starting and ending points that are associated with declared variables. With instant vertex elimination, however, we can do nested differentiation, as each variable is associated with a unique vertex in the Life-DAG, temporaries having been instantly eliminated, and we may traverse the graph from any vertex to any other, at any stage during the computation of the function f.

Let us consider a function `eval_residue(y, x)` that for given vectors `x` and `y` computes the implicit function $G_x(y)$. Then the code in Table 2.2 illustrates

how ADTAGEO computes the solution of $G_x(y) = 0$ using the Newton method and then at the end computes the derivative $\frac{dy}{dx}$ at the solution point. For the example function in Table 2.3 the results obtained for several given values of **x** are shown in Table 2.4.

Table 2.2: Nested differentiation

```
#include <iostream>
#include <iomanip>
#include <daglad.hpp>                          // defines active datatype
#include <daglad_propagator.hpp>               // defines propagator type

using namespace std;
using namespace ADTAGEO;

extern daglad eval_residue(const daglad& y, const daglad& x);
double epsilon = 1E-12;

void newton(const daglad& x, daglad& y) {
     int iter = 0;
     daglad r;                                 // residue
     daglad t;                                 // step
     do {
        iter++;
        r = eval_residue(y, x);                // evaluate residue r = G_x(y)
        daglad_propagator dp(P_REVERSE);       // reverse propagator for derivatives
        dp.seed(r,1.0);                        // r̄ ← 1.0
        double deriv = dp % y;                 // propagate reverse
                                               // deriv = ȳ = r̄ dG_x/dy (y)
        t = r / deriv;                         // newton step
        y = y - t;                             // correction
     } while ( fabs(t.val()) > epsilon );      // until step is small enough
     cout << "Newton method required " << iter << " iterations" << endl;
}

int main() {
    daglad x, y = 1.0;                         // initial value
    double xv;
    cout << "Input the point x" << endl;       // get point x
    cin >> xv;
    x = xv;

    newton(x, y);                              // evaluate y = f(x) at given x
    daglad_propagator df(P_FORWARD);           // forward propagator for derivatives
    df.seed(x,1.0);                            // ẋ ← 1.0
    double deriv = df % y;                     // propagate forward
                                               // deriv = ẏ = df/dx (x) ẋ
    cout << "y = f(" << x.val() << ") = " << y.val() << endl;
    cout << "dy/dx = " << deriv << endl;
    return 0;
}
```

ADTAGEO should also allow us to compute second order derivatives of f as Hessian vector products $\bar{y}f''\dot{x}$. The Life-DAG has to then store local Hessians too, and second order elimination rules have to be applied during the vertex elimination. Moreover, two-way propagation has to be done: first send the tangent $\dot{x}$ from starting vertices towards the ending nodes, followed by a reverse sweep that carries now the initial adjoint $\bar{y}$ together with a second order propagation of the already computed tangent $\dot{y}$.

Table 2.3: Example implicit function

```
daglad eval_residue(const daglad& y, const daglad& x) {
       daglad f = sin(exp(y)) - cos(exp(y)) * x * x;
       return f;
}
```

Table 2.4: Results of nested differentiation

```
> ./a.out
Input the point x
-0.5
Newton method required 5 iterations
y = f(-0.5) = 1.21982
dy/dx = -0.277914
> ./a.out
Input the point x
-0.4
Newton method required 5 iterations
y = f(-0.4) = 1.194
dy/dx = -0.236355
> ./a.out
Input the point x
-0.3
Newton method required 5 iterations
y = f(-0.3) = 1.1729
dy/dx = -0.184189
> ./a.out
Input the point x
-0.2
Newton method required 5 iterations
y = f(-0.2) = 1.15738
dy/dx = -0.125523
> ./a.out
Input the point x
-0.1
Newton method required 5 iterations
y = f(-0.1) = 1.14791
dy/dx = -0.0634536
> ./a.out
Input the point x
0.0
Newton method required 5 iterations
y = f(0) = 1.14473
dy/dx = 0
```

```
> ./a.out
Input the point x
0.1
Newton method required 5 iterations
y = f(0.1) = 1.14791
dy/dx = 0.0634536
> ./a.out
Input the point x
0.2
Newton method required 5 iterations
y = f(0.2) = 1.15738
dy/dx = 0.125523
> ./a.out
Input the point x
0.3
Newton method required 5 iterations
y = f(0.3) = 1.1729
dy/dx = 0.184189
> ./a.out
Input the point x
0.4
Newton method required 5 iterations
y = f(0.4) = 1.194
dy/dx = 0.236355
> ./a.out
Input the point x
0.5
Newton method required 5 iterations
y = f(0.5) = 1.21982
dy/dx = 0.277914
> ./a.out
Input the point x
0.6
Newton method required 5 iterations
y = f(0.6) = 1.24908
dy/dx = 0.304639
```

Chapter 3

The total quasi-Newton method

In this chapter we discuss a method for the solution of a general nonlinear programming problem for the form

$$\min_{x \in \mathbb{R}^n} f(x), \quad f \in C^1 \tag{3.1a}$$

$$\text{s.t. } c_i(x) = 0, \quad c_i \in C^1,\ i = 0 \dots m_{\text{eq}} - 1 \tag{3.1b}$$

$$c_j(x) \leqslant 0, \quad c_j \in C^1,\ j = m_{\text{eq}} \dots m_{\text{up}} - 1 \tag{3.1c}$$

where $x \in \mathbb{R}^n$ are the optimization variables, $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function, and $c : \mathbb{R}^n \longrightarrow \mathbb{R}^{m_{\text{up}}}$ are the constraints, out of which the first m_{eq} are equality constraints and the rest are inequality constraints. The first-order necessary conditions for optimality can be written for once continuously differentiable functions f and c, however each of f and c_i must be assumed to be in C^2 in order to be able to write the second-order sufficiency conditions for optimality. In general the number of variables n is quite large compared to the number of constraints m_{up} but we do not make this formal assumption.

3.1 Optimality and Newton's Method

If x^* is a local solution of (3.1) where the gradients of active constraints are linearly independent, thus satisfy the linear independence constraint qualification (LICQ), then there exists a unique vector of nonnegative lagrange multipliers $\lambda \in \mathbb{R}^{m_{\text{up}}}$ such that stationarity, feasibility and complementarity hold [NOCEDAL & WRIGHT, 1999], i.e.

$$0 = \nabla f(x^*) + \lambda^\top \nabla c(x^*) \tag{3.2a}$$

$$0 = c_i(x^*),\ 0 \leqslant i < m_{\text{eq}} \tag{3.2b}$$

$$0 \geqslant c_j(x^*),\ m_{\text{eq}} \leqslant j < m_{\text{up}} \tag{3.2c}$$
$$0 = \lambda_i c_i(x^*),\ 0 \leqslant i < m_{\text{up}} \tag{3.2d}$$

The above are the first-order necessary conditions for stationarity. The last equation (complementarity) is called strict whenever it holds along with

$$|c_i(x^*)| + |\lambda_i| \neq 0 \quad \text{for } 0 \leqslant i < m_{\text{up}}$$

The second-order sufficiency conditions for minimality may be written for twice continuously differentiable functions f and c as

$$s^\top [\nabla^2 f(x^*) + \lambda^\top \nabla^2 c(x^*)] s \geqslant 0 \quad \forall s \text{ s.t. } s^\top \nabla c_i(x^*) = 0 \text{ for each } c_i(x^*) = 0$$

A good method to solve for the optimality conditions in (3.2) is the Newton's method with an active set strategy [NOCEDAL & WRIGHT, 1999]. This method defines a set of active constraints at the current point and then minimizes the objective in the tangent space of the active constraints. The convergence of Newton's method depends on the starting point being close to the local minimum, however globalization may be achieved using line-searches [NOCEDAL & WRIGHT, 1999, Chapter 3] or trust region aproaches [CONN et al., 2000].

In order to write the method more concretely we first define a working index set

$$\mathcal{I}(x) \subset \{0, \ldots, m_{\text{up}} - 1\} \text{ with } |\mathcal{I}(x)| = m \geqslant m_{\text{eq}} \text{ and } i \in \mathcal{I}(x)\, \forall 0 \leqslant i < m_{\text{eq}}$$

The constraints whose indices lie in $\mathcal{I}$ are considered active at the current point x and we define the permutation mapping $\pi_x : \{0, \ldots, m_{\text{up}} - 1\} \longrightarrow \{0, \ldots, m_{\text{up}} - 1\}$ such that

$$0 \leqslant \pi_x(i) < m \iff i \in \mathcal{I}(x) \tag{3.3}$$

It is quite clear that $m \leqslant n$ must hold in order for LICQ to be satisfied by the active constraints.

At any point given x the working index set $\mathcal{I}(x)$ is implicitly defined by defining the corresponding permutation mapping π_x and we permute the vector of constraint functions along with the vector of corresponding lagrange multipliers. We denote by $c_{\mathcal{I}(x)}$ and $\lambda_{\mathcal{I}(x)}$ the first m elements of these permuted vectors. An exact active set means that the rest of the $m_{\text{up}} - m$ constraint functions are satisfied strictly at the current point.

For a strict active set the minimization problem (3.1) reduces to an equality constrained problem on the tangent space of the active constraints

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t. } c_{\mathcal{I}(x)}(x) = 0$$

We define the Lagrangian function as

$$\mathcal{L}(x, \lambda) = f(x) + \lambda_{\mathcal{I}(x)}^\top c_{\mathcal{I}(x)}(x)$$

and solve for the point (x, λ) that satisfies the first-order necessary condition for stationarity

$$\nabla_{x,\lambda} \mathcal{L}(x, \lambda) = 0$$

The above expands to the following set of equations called the Karush-Kuhn-Tucker equations

$$\nabla f(x) + \lambda_{\mathcal{I}(x)}^\top \nabla c_{\mathcal{I}(x)}(x) = 0 \tag{3.4a}$$

$$c_{\mathcal{I}(x)}(x) = 0 \tag{3.4b}$$

At any iterate (x_k, λ_k) of the Newton's method a direction of descent is found by solving

$$\begin{bmatrix} \nabla^2 f(x_k) + \lambda_{k\mathcal{I}(x_k)}^\top \nabla^2 c_{\mathcal{I}(x_k)}(x_k) & \nabla c_{\mathcal{I}(x_k)}(x_k)^\top \\ \nabla c_{\mathcal{I}(x_k)}(x_k) & 0 \end{bmatrix} \begin{bmatrix} s_k \\ \lambda_{k+1\mathcal{I}(x_k)} \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -c_{\mathcal{I}(x_k)}(x_k) \end{bmatrix}$$

3.2 Total quasi-Newton Method

A well known approach in literature is to approximate the second-order derivative matrices by using a starting matrix B_0 and updating it in each step by low-rank updates. Two such updates are the SR1 (symmetric rank-1) update and BFGS (named after Broyden, Fletcher, Goldfarb and Shanno) update [NOCEDAL & WRIGHT, 1999, Chapter 8]

$$B_+ = B + \beta \frac{(w - Bs)(w - Bs)^\top}{(w - Bs)^\top s} \qquad \text{(SR1)}$$

$$B_+ = B - \frac{Bss^\top B}{s^\top Bs} + \frac{ww^\top}{w^\top s} \qquad \text{(BFGS)}$$

satisfying the secant condition $w = B_+ s$.

This is now a quasi-Newton methods as the direction of descent is a approximation of the one obtained from Newton's method. The updates consist of vectors that may be computed as Hessian-vector products and gradients, that are computationally much cheaper to compute using algorithmic differentiation. As algorithmic differentiation also allows one to compute Jacobian-vector products in a similar efficient manner, it is a short step ahead that one may replace the Jacobian matrix of the active constraints by an approximation that is similarly updated by low-rank updates. The method is then called the total quasi-Newton methods. Two such updates for the Jacobian matrix are TR1 (two sided rank-1) update and the adjoint Broyden update [SCHLENKRICH et al., 2009a,b, 2010]

$$A_+ = A + \frac{(y + As)(\bar{y}^\top + \sigma^\top A)}{(\bar{y}^\top + \sigma^\top A)s} \qquad \text{(TR1)}$$

$$A_+ = A + \frac{\sigma\sigma^\top}{\sigma^\top\sigma}(\nabla c(x_+) - A) \qquad \text{(ADJBR)}$$

satisfying the direct secant and adjoint tangent conditions $y = A_+ s$ and $\bar{y}^\top = \sigma^\top A$.

This means we solve for the descent direction using a totally approximated system

$$\begin{bmatrix} B_k & A_{k\,\mathcal{I}(x_k)}^\top \\ A_{k\mathcal{I}(x_k)} & 0 \end{bmatrix} \begin{bmatrix} s_k \\ \lambda_{k+1\mathcal{I}(x_k)} \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -c_{\mathcal{I}(x_k)}(x_k) \end{bmatrix}$$

GRIEWANK et al. [2007a] have developed an algorithm which enables one to update a QR based factorization of the above matrix. The constraint Jacobian is QR-factorized as

$$A = [L \;\; 0][Y \;\; Z]^\top = LY^\top$$

Here, $[Y \;\; Z]$ is the orthogonal Q-factor with $YY^\top + ZZ^\top = \mathbb{I}$. The columns of Y span the range space of the constraint Jacobian and the columns of Z span the nullspace of the constraint Jacobian. Now the factorized system looks like

$$\begin{bmatrix} E & C & L^\top \\ C^\top & UU^\top & 0 \\ L & 0 & 0 \end{bmatrix} \begin{bmatrix} Y^\top s \\ Z^\top s \\ \lambda_{+\mathcal{I}(x)} \end{bmatrix} = \begin{bmatrix} -Y^\top \nabla f(x) \\ -Z^\top \nabla f(x) \\ -c_{\mathcal{I}(x)}(x) \end{bmatrix} \tag{3.5}$$

with $E = Y^\top BY$, $C = Y^\top BZ$ and $UU^\top = Z^\top BZ$. The update formulae are then reformulated to update the factors directly. Figure 3.1 sketches the process of factorised updates in terms of the martix-vector operations required, for full

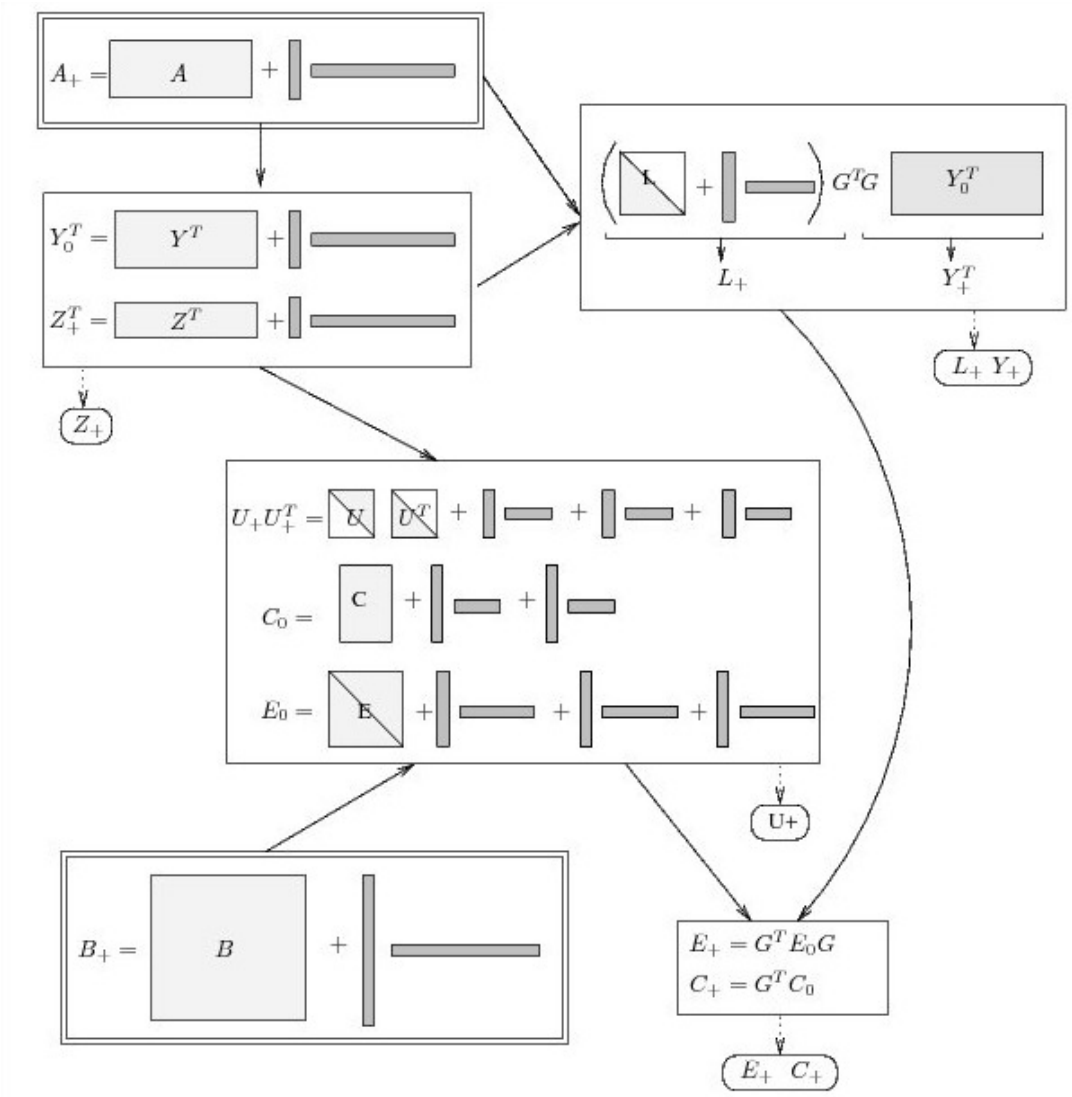


Figure 3.1: Update scheme for the QR based factorisation of the KKT-matrix

details refer GRIEWANK et al. [2007a].

The memory setup is such that the activity of the constraints can be tracked by permuting the rows of A by the permutation π_x via low-rank updates too. Furthurmore, a limited-memory version of the updates that is free of the null-space representation has also been developed by [BOSSE et al., 2009; BOSSE, 2009; SCHLOSSHAUER, 2009].

Since the convergence rate of the (quasi-) Newton method depends on the choice of the starting point, globalization strategies are needed in order to solve application problems. One such strategy is to use line-search on an appropriate merit function. Since the evaluation of a function value and its directional derivative can be done cheaply at any point using algorithmic differ-

entiation, line-searches based on cubic interpolation of the merit function can be used. One such well known line-search algorithm is the Zoom algorithm of NOCEDAL & WRIGHT [1999, Algorithm 3.3]. A Shift and Zoom algorithm has also been recently developed [KREITERLING, 2007] (see Table 3.2). Several choices for the Merit function also appear in literature, like an augmented Lagrangian function, doubly augmented Lagrangian or an l^1-penalty function [BOSSE, 2009].

As a part of the MATHEON research center of the German Research Foundation GRIEWANK et al. [2004–2010] developed a software tool `LRAMBO` based on total quasi-Newton method with updates on factorizations as described above. This is a `C++` library with modular design and the optimisation algorithm is sketched in Table 3.1.

Set initial point x_0, λ_0 and $k = 0$
do {
  do {
    Solve approximating KKT system (3.5)
    If descent in meritfunction **break**
    Update A_k by appropriate low rank updates
    If test for inequalities fail, change active set
    Iterate $k = k + 1$
    } **until**(descent and no change in active set)
  Linesearch $\rightarrow$ Step multiplier α_k
  Set $x_{k+1} = x_k + \alpha_k s_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \sigma_k$
  Change active set
  Update A_k by appropriate low rank updates
  Update B_k by SR1/BFGS
  Iterate $k = k + 1$
} **while**(not solved)

Table 3.1: Scheme of the total quasi-Newton method - The solver package `LRAMBO` includes a practical `C++` implementation

In order to use the software package `LRAMBO` one only needs to overload an abstract `C++` class that implements methods to compute the function value $f(x)$, constraints $c(x)$, the gradient $\nabla f(x)$ and the Jacobian-vector products $\nabla c_{\mathcal{I}(x)}(x)s$ or $\sigma^\top \nabla c_{\mathcal{I}(x)}(x)$ at a given point x and permutation π_x. A derivative evaluation interface using `ADOL-C` is also included and one then need not implement any derivative evaluation methods. Both dense and limited memory

Set the initial interval $[\alpha_l\,,\ \alpha_r]$ and upperbound $\alpha_{\max} > \alpha_l$
DO
Determine a Hermitian interpolation $P(t)$, using the function values at the interval boundaries α_r, α_l
IF it exists, compute the minimiser t_* of cubic polynomial $P(t)$
ELSE set $t_* = -\infty$
IF the function value at right interval side α_r is bigger than on the left side α_l
THEN reduce upperbound $\alpha_{\max} = \alpha_r$
Compute the predicted: $P(\alpha_l) - P(t_*)$ and actual reduction: $P(\alpha_l) - P(\alpha_r)$
IF the minimiser t_* lies outside the interval $(\alpha_l\,,\ \alpha_{\max}]$ and the slope on the right interval boundary promise further reduction
THEN Shift interval further right: $\alpha_l = \alpha_r$ and $\alpha_r = \min(\alpha_{\max}, \alpha_l + \delta)$
ELSE IF $t_* > \alpha_r$ but the actual compared to the predicted reduction is to small and the slope on the right interval boundary promise further reduction
THEN Shift interval further right: $\alpha_l = \alpha_r$ and $\alpha_r = \min(\alpha_{\max}, \alpha_l + \delta)$
ELSE IF the interval boundary slopes indicate the lack of positive curvature and on the right interval boundary further reduction is promised
THEN Shift interval further right: $\alpha_l = \alpha_r$ and $\alpha_r = \min(\alpha_{\max}, \alpha_l + \delta)$
ELSE IF the minimizer is within expected interval, i.e. $t_* \in (\alpha_l, \alpha_{\max}]$, but the actual compared to the predicted reduction is insfficient
THEN Perform a Zoom operation, i.e. set the interval $\alpha_r = t_*$
ELSE accept the step and **RETURN** Step length α_r **END**
WHILE(NO SUFFICIENT DESCENT)

Table 3.2: Shift and Zoom Linesearch

matrix updates are available for use. There are also interfaces for providing the problem to be solved in the `.sif` format of the CUTEr library [CONN et al., 2002] as well as calling the optimization routine from within Matlab®. For further information on `LRAMBO` please refer `http://www.math.hu-berlin.de/~griewank/C12/`.The source code may be obtained with express permission of the authors from the following `git`-Repository `http://www.math.hu-berlin.de/~lorambo/rambo.git`. The software has been tested to work on Linux and Mac-OS X environments. An interface to use `LRAMBO` as the solver also exists in the `NEOS` server at `http://neos.mcs.anl.gov/neos/solvers/nco:LRAMBO/C.html`.

Chapter 4

Modelling of a single beam

4.1 The state variables

The state of any beam in the loaded structure shall be determined by specifying the positions and the orientations of both its ends. Each end-piece is represented as a point object in space with six degrees of freedom, a position vector with three components, and a *quaternion* of four components whose euclidian length is 1. So we shall have for each beam a couple of position vectors $p_1, p_2 \in \mathbb{R}^3$ and a couple of orientation vectors $Q_1, Q_2 \in S^3$, where $S^3 \subset \mathbb{R}^4$ is the unit sphere in four dimensional space. These state variables along with the design variables define the beam energy $\mathcal{E} = \mathcal{E}(p_2 - p_1, Q_1, Q_2)$.

The orientation vectors Q_1 and Q_2 belong to the non commutative Quaternion Algebra $\mathbb{H}(\mathbb{R}^4)$, so that in general $Q_1 Q_2 \neq Q_2 Q_1$. On the quaternion unit sphere $Q \in \mathbb{H}(S^3)$, i.e. for all unit length quaternions, the inverse is defined by its conjugate quaternion Q^*. Such quaternions represent a general rotation of a vector in three dimensional space [GOLDSTEIN, 1973]. According to Euler's theorem of rigid body rotations every orthogonal matrix with unit determinant describes a right hand rotation. Each unit quaternion can be used to determine a unique rotation matrix.

$$
\begin{aligned}
&[Q^w, Q^x, Q^y, Q^z] \mapsto \\
&\begin{bmatrix}
-Q^{y2} + Q^{x2} - Q^{z2} + Q^{w2} & 2(Q^x Q^y - Q^z Q^w) & 2(Q^x Q^z + Q^y Q^w) \\
2(Q^x Q^y + Q^z Q^w) & Q^{y2} - Q^{x2} - Q^{z2} + Q^{w2} & 2(Q^y Q^z - Q^x Q^w) \\
2(Q^x Q^z - Q^y Q^w) & 2(Q^y Q^z + Q^x Q^w) & -Q^{y2} - Q^{x2} + Q^{z2} + Q^{w2}
\end{bmatrix}
\end{aligned}
\tag{4.1}
$$

This nonlinear action of a quaternion on three dimensional vectors will be denoted by the symbol $\circ$ in the rest of this document. This kind of description of orientation has been used for designing singularity free algorithms for molecular dynamics and crystal lattices [EVANS & MURAD, 1977; GRIEWANK et al.,

1979]. The advantage of using quaternions for orientations is that this representation is avoids the singular plane that is encountered with Euler angles GOLDSTEIN [1973], which has disastrous effects on numerical computations. Quaternions have local isometry with respect to the rotation angle between orientations. They are also more compact and faster than matrices.

The quaternions Q_1 and Q_2 for a beam are defined such that their action on the basis unit vector $\mathbf{e}_z$ of $\mathbb{R}^3$ rotates it to the tangential direction vector at the end points p_1 and p_2 of the beam, i.e. $Q_1 \circ \mathbf{e}_z$ is the tangential direction vector at p_1 and $Q_2 \circ \mathbf{e}_z$ at p_2. At the same time the two other basis vectors $\mathbf{e}_x$ and $\mathbf{e}_y$ are also rotated by the quaternions in the directions of the major and minor axes of the elliptic cross-section of the beam.

4.2 State parameterisation

For any parameterisation of the position vector $\wp(t)$ of the central line of the beam, its deformed length $\widetilde{L}$ is given by

$$\widetilde{L} = \int_0^T r(t)\mathrm{d}t$$

Here $\|\wp'(t)\| = r(t)$ and $\wp'(t) = r(t)[\mathfrak{P}(t) \circ \mathbf{e}_z]$, where $\mathfrak{P}(t)$ is a quaternion function that describes the orientation of the beam at t. The elastic energy stored in the material due to change in length is given by

$$\mathcal{E}_l = \frac{EA}{2}\frac{(L-\widetilde{L})^2}{L^2} \tag{4.2a}$$

where L is the undeformed length and A the average area of crossection, and that due to its bending is given by

$$\mathcal{E}_b = \frac{E}{2}\int_0^T I(t)\frac{\|\wp''(t)\|^2}{r(t)^4}\mathrm{d}t \tag{4.2b}$$

E is the Young's modulus and $I(t)$ is the area moment of inertia of the cross-section at t. Using the quaternion function $\mathfrak{P}(t)$ the torsional part of the elastic energy can also be written as

$$\mathcal{E}_t = \frac{GA}{2}\int_o^T (\mathfrak{P}'(t) \circ \mathbf{e}_x)\mathrm{d}t \tag{4.2c}$$

G being the shear modulus of elasticity.

In order to determine the complete state of the beam we choose the parameter t to vary along the straight line joining the two end points. Let the distance between the two endpoints be Z, we first rotate and translate the coordinate system for each beam such that

1. both the end points lie on the z-axis, one on the origin, putting the other at the point $(0, 0, Z)$,
2. two mutually orthogonal unit vectors selected to form a right handed orthonormal coordinate system w.r.t. the z-axis form the x- and y-axes.

Let us first consider the case where no torsion of the beam is present. Then the complete state of the beam is determined by the central curve $\wp(z)$ of the beam, where $z \in [0, Z]$.

$$\wp(z) = \begin{bmatrix} x(z) \\ y(z) \\ z \end{bmatrix}$$

The tangent at any point is in general given by a quaternion function $\mathfrak{P}(z)$ such that:

$$\wp'(z) = r(z)\left[\mathfrak{P}(z) \circ \mathbf{e}_z\right]$$

where $r(z)$ is the length of the tangent vector.

$$r(z)^2 = 1 + x'(z)^2 + y'(z)^2 \tag{4.3}$$

The above quaternion function can also be used to determine the extent of torsion in the beam as an action on $\mathbf{e}_x$ or $\mathbf{e}_y$. In the torsion free case, however, we need only the tangent, which can also be written as

$$\wp'(z) = \begin{bmatrix} x'(z) \\ y'(z) \\ 1 \end{bmatrix}$$

This formulation limits the deformation of the beam to be a *function* of the line joining its end points, i.e. small deformations. The tangents are always assumed to have a positive component in the direction of the z-axis. Large deformations and loops are not allowed.

4.3 The state functions

Assuming small deformation in state, i.e. $|x(z)| \ll Z$ and $|y(z)| \ll Z$ for each $z \in [0, Z]$, the functions $x(z)$ and $y(z)$ are the solutions of the approximate state

equations

$$\begin{aligned} EI_x \frac{\mathrm{d}^4 x}{\mathrm{d}z^4} + P\frac{\mathrm{d}^2 x}{\mathrm{d}z^2} &= 0 \\ EI_y \frac{\mathrm{d}^4 y}{\mathrm{d}z^4} + P\frac{\mathrm{d}^2 y}{\mathrm{d}z^2} &= 0 \end{aligned} \tag{4.4}$$

along with the boundary conditions provided by the given endpoints and tangent directions. Here E is the Young's modulus of the material and I_x and I_y are the area moments of inertia of the cross-section about the x- and y-axes, and P is a compressional force acting along the beam's central axis, which is assumed to be constant along the beam. The characteristic polynomials for both the above 4^{th} order ordinary differential equations with constant coefficients are of the form

$$\chi_x(\lambda) = \lambda^2(EI_x\lambda^2 + P) \quad \text{and} \quad \chi_y(\lambda) = \lambda^2(EI_y\lambda^2 + P)$$

The roots of these characteristic polynomials and thus the solution of (4.4) depend on value of P, forming three separate elastic phases. If $P > 0$, when the end points are being pushed towards each other, we have two imaginary roots of the characteristic polynomial and a double root at 0. The fundamental solutions are, thus, trigonometric functions, a linear function and a constant. This is the compressed bending phase. On the other hand if $P < 0$, when the ends are being pulled apart, we have two real roots of the characteristic polynomial and a double root at 0. The fundamental solutions are thus exponential functions, a linear function and constant. This is the extensional bending phase. For the case of $P = 0$, when the ends are neither being pulled nor pushed, the characteristic polynomial has a zero of 4th order at 0 and we have a cubic polynomial as the solution. This is the pure bending phase. So we have the following as the fundamental solution system of the linear boundary value problem of 4th order [WALTER, 2000].

$$\begin{aligned} P > 0 \implies & \begin{cases} x(z) &= A_x \sin k_x z + B_x \cos k_x z + C_x z + D_x \\ y(z) &= A_y \sin k_y z + B_y \cos k_y z + C_y z + D_y \end{cases} \\ P < 0 \implies & \begin{cases} x(z) &= A_x \exp(k_x z) + B_x \exp(-k_x z) + C_x z + D_x \\ y(z) &= A_y \exp(k_y z) + B_y \exp(-k_y z) + C_y z + D_y \end{cases} \\ P = 0 \implies & \begin{cases} x(z) &= A_x z^3 + B_x z^2 + C_x z + D_x \\ y(z) &= A_y z^3 + B_y z^2 + C_y z + D_y \end{cases} \end{aligned} \tag{4.5}$$

Here, the constants A_x, B_x, C_x, D_x and A_y, B_y, C_y, D_y are all determined by using the given endpoint positions and tangents as boundary conditions, and k_x, k_y are defined by

$$k_x^2 \equiv \frac{|P|}{EI_x} \quad k_y^2 \equiv \frac{|P|}{EI_y}$$

The unknown P needs to be determined, and we solve for it from the following equation for the change in length, which is straightforward from the Hooke's Law that relates the change in length to the applied compressional or tensile force [LOVE, 1944].

$$L - \int_0^Z \sqrt{x'(z)^2 + y'(z)^2 + 1}\,\mathrm{dz} = \frac{PL}{EA} \tag{4.6}$$

with A being the area of cross-section of the beam. In the case of small deflection the integral term can be approximated as

$$\int_0^Z \sqrt{x'(z)^2 + y'(z)^2 + 1}\,\mathrm{dz} \approx Z + \frac{1}{2}\int_0^Z (x'(z)^2 + y'(z)^2)\mathrm{dz}$$

The Hooke's law equation (4.6) in this form is a non-linear equation. The integrand in the region close to $P = 0$ has three separate phases, and if we want to solve this using an iterative method, we would need to jump between the three different phases of the solution (4.5) to the state equation. This makes it difficult for a derivative based solution method like Newton method to solve the equation and give meaningful results. Thus, we need to reformulate the solutions of the state equations in a stable and continuous form w.r.t. the parameter P by changing the form of the fundamental solutions of (4.4). For this purpose we consider the two power series S_2 and S_3 defined by

$$S_3(\tilde{k}, z) = z^3 \left[\sum_{j=0}^{\infty} \frac{(-1)^j \tilde{k}^j z^{2j}}{(2j+3)!}\right]$$

$$S_2(\tilde{k}, z) = z^2 \left[\sum_{j=0}^{\infty} \frac{(-1)^j \tilde{k}^j z^{2j}}{(2j+2)!}\right] \quad = \frac{\partial S_3}{\partial z}(\tilde{k}, z)$$

with

$$\tilde{k} = \frac{P}{EI}$$

Here, E is the Young's modulus and I the area moment of inertia about the direction of interest. For the sake of simplicity in the following, I is assumed to be the same in both directions x and y. The two series can be derived as the

solutions of the state equations (4.4) by the Frobenius method [HEUSER, 2004]. For $P > 0$ these two series represent a linear combination of sine and cosine functions with a linear and constant function respectively.

$$S_3(\tilde{k}, z) = \frac{kz - \sin kz}{k^3}$$

$$S_2(\tilde{k}, z) = \frac{1 - \cos kz}{k^2}$$

where $k^2 = \tilde{k}$. On the other hand for $P < 0$ the two series represent a linear combination of the hyperbolic sine and cosine functions with a linear and constant function respectively.

$$S_3(\tilde{k}, z) = \frac{\sinh kz - kz}{k^3}$$

$$S_2(\tilde{k}, z) = \frac{\cosh kz - 1}{k^2}$$

For $P = 0$ the series both reduce to $\frac{z^3}{6}$ and $\frac{z^2}{2}$ respectively. Hence we can write the modified state functions for all values of P as follows.

$$\begin{aligned} x(z) &= A_x S_3(\tilde{k}_x, z) + B_x S_2(\tilde{k}_x, z) + C_x z + D_x \\ y(z) &= A_y S_3(\tilde{k}_y, z) + B_y S_2(\tilde{k}_y, z) + C_y z + D_y \end{aligned}$$

As the series converge quite fast, we can evaluate them to machine precision with a moderate number of terms using the following modified Horner scheme.

$$S_2(\tilde{k}, z) = \frac{z^2}{1 \times 2}\left(1 - \frac{\tilde{k}z^2}{3 \times 4}\left(1 - \frac{\tilde{k}z^2}{5 \times 6}\left(1 - \ldots\right)\right)\right)$$

$$S_3(\tilde{k}, z) = \frac{z^3}{2 \times 3}\left(1 - \frac{\tilde{k}z^2}{4 \times 5}\left(1 - \frac{\tilde{k}z^2}{6 \times 7}\left(1 - \ldots\right)\right)\right)$$

For the purpose of computation we compute this modified Horner scheme as a function of one variable $\varkappa = \tilde{k}z^2$ without the leading factor z^2 or z^3, which is then multiplied at the end. This simplifies the derivative computation.

In general the modified Horner scheme for a series in $\varkappa$ can be given as

$$\sum_{j=0}^{\infty} \frac{a_j b^{j+c}}{(2j+d)!} \varkappa^j =$$

$$\frac{b^c}{d!}\left[a_0 + \frac{b\varkappa}{(d+1)(d+2)}\left(a_1 + \ldots\left(a_j + \frac{b\varkappa}{(d+2j+1)(d+2j+2)}\left(\ldots\right)\right)\right)\right]$$

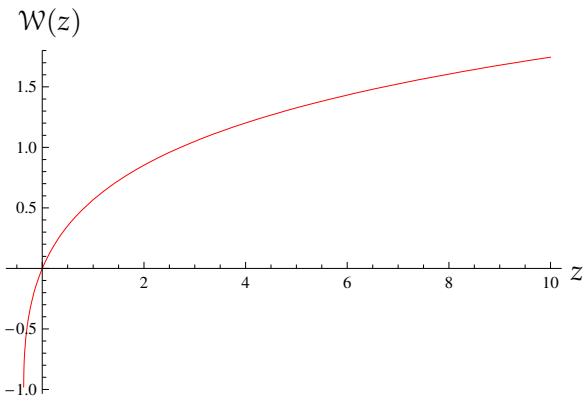


Figure 4.1: A single valued restriction of the Lambert $\mathcal{W}$ function

The j-th term of the series can be bounded approximately using Stirling's formula as

$$\max(|S_3^j|, |S_2^j|) \leqslant \left| \frac{(\tilde{k}z^2)^j}{(2j)!} \right| \approx \frac{1}{\sqrt{4\pi j}} \left| \frac{\tilde{k}e^2z^2}{4j^2} \right|^j$$

The maximal value of j needed to bound the exponential factor by any $\varepsilon > 0$ can be written as

$$j_{\max} = -\frac{1}{2} \ln \varepsilon \left[\mathcal{W} \left(\frac{-\ln \varepsilon}{\sqrt{|\tilde{k}e^2z^2|}} \right) \right]^{-1}$$

Here $\mathcal{W}$ is the Lambert $\mathcal{W}$ function [CORLESS et al., 1996] also called the product log, that represents the complex multivalued inverse of $f(w) = we^w \forall w \in \mathbb{C}$. Figure 4.1 shows the graph of $\mathcal{W}_0$, a single valued restriction of $\mathcal{W}$, with the domain restricted to $[-1/e, \infty)$ and the range to $[-1, \infty)$.

An asymptotic formula [DE BRUIJN, 1981; CORLESS et al., 1996] that yields reasonably accurate results for $z \gtrsim 3$ is

$$\begin{aligned} \mathcal{W}(z) &\approx \ln z - \ln \ln z + \sum_{k=0}^{\infty} \sum_{m=0}^{\infty} \binom{m}{k} (\ln \ln z)^{m+1} (\ln z)^{-k-m-1} \\ &= L_1 - L_2 + \frac{L_2}{L_1} + \frac{L_2(L_2-2)}{2L_1^2} + \mathrm{O}\left[\left(\frac{L_2}{L_1} \right)^3 \right] \end{aligned}$$

where $L_1 = \ln z$ and $L_2 = \ln \ln z$.

The computation of the deformed length and energy, however, with these truncated series expansions of S_2 and S_3 requires a discretised integration procedure like Gauss integration as we cannot utilise the orthogonality of the two functions while integrating on the domain $[0, Z]$. Thus, we have some cross

terms in the integrand, which make analytical integration very difficult. The truncation error along with the discretisation error in such a discretised evaluation, however, is large enough to give physically erroneous results in some cases.

4.4 Series expansions

In order to be able to compute the deformed length and bending energy to a high accuracy we should formulate closed form expressions for these and later use the series expansions whenever necessary. To this end we translate our local coordinate system so that $z = 0$ lies at the centre of the beam and the endpoints are at $z = -Z/2$ and $z = Z/2$. We can then rewrite the two series solutions S_3 and S_2 of the state equation (4.4) that vanish at the boundary of the interval $[-\frac{Z}{2}, \frac{Z}{2}]$ as an odd and an even power series.

$$S_{\text{odd}}(\tilde{k}, z) = z \sum_{j=0}^{\infty} \left[\frac{(-1)^j \tilde{k}^j}{(2j+3)!} \left\{ \left(\frac{Z}{2} \right)^{2j+2} - z^{2j+2} \right\} \right] \tag{4.7}$$

$$S_{\text{even}}(\tilde{k}, z) = \sum_{j=0}^{\infty} \left[\frac{(-1)^j \tilde{k}^j}{(2j+2)!} \left\{ \left(\frac{Z}{2} \right)^{2j+2} - z^{2j+2} \right\} \right] \tag{4.8}$$

These series converge to the following trigonometric, hyperbolic or polynomial functions depending on the value of $\tilde{k}$

$$S_{\text{odd}}(\tilde{k}, z) = \begin{cases} \frac{1}{k^3} \left[\sin kz - \frac{2}{Z} z \sin \frac{kZ}{2} \right] & \text{if } \tilde{k} > 0 \; k^2 = \tilde{k} \\ \frac{1}{k^3} \left[\sinh kz - \frac{2}{Z} z \sinh \frac{kZ}{2} \right] & \text{if } \tilde{k} < 0 \; k^2 = -\tilde{k} \\ z \frac{Z^2}{24} - \frac{z^3}{6} & \text{if } \tilde{k} = 0 \end{cases} \tag{4.9}$$

$$S_{\text{even}}(\tilde{k}, z) = \begin{cases} \frac{1}{k^2} \left[\cos kz - \cos \frac{kZ}{2} \right] & \text{if } \tilde{k} > 0 \; k^2 = \tilde{k} \\ \frac{1}{k^2} \left[\cosh kz - \cosh \frac{kZ}{2} \right] & \text{if } \tilde{k} < 0 \; k^2 = -\tilde{k} \\ \frac{Z^2}{8} - \frac{z^2}{2} & \text{if } \tilde{k} = 0 \end{cases} \tag{4.10}$$

The boundary conditions on the slopes at the endpoints $z = -Z/2$ and $z = Z/2$ can then be used to determine the coefficients B_{odd} and B_{even} such that the

displacement at z along each of the directions of $\xi = x$ or y is simply given by

$$\xi(z) = B^{\xi}_{\text{odd}} S_{\text{odd}}(\tilde{k}, z) + B^{\xi}_{\text{even}} S_{\text{even}}(\tilde{k}, z)$$

with the coefficients suitably computed for that direction ξ.

The two terms in the displacement function are now orthogonal in the domain $[-Z/2, Z/2]$. Thus, the integrals involved in computation of deformed length and elastic energy can be computed by integrating the even and the odd terms separately. The cross terms vanish due to the orthogonality. Closed form expressions for the deformed length and elastic energy can now be derived as either power series or combinations of trigonometric or hyperbolic functions. Given slopes t^{ξ}_1 and t^{ξ}_2 in directions $\xi = x$ or y at the two endpoints we solve

$$\begin{bmatrix} S'_{\text{odd}}(\tilde{k}, -\frac{Z}{2}) & S'_{\text{even}}(\tilde{k}, -\frac{Z}{2}) \\ S'_{\text{odd}}(\tilde{k}, \frac{Z}{2}) & S'_{\text{even}}(\tilde{k}, \frac{Z}{2}) \end{bmatrix} \begin{bmatrix} B^{\xi}_{\text{odd}} \\ B^{\xi}_{\text{even}} \end{bmatrix} = \begin{bmatrix} t^{\xi}_1 \\ t^{\xi}_2 \end{bmatrix}$$

with

$$S'_{\text{odd}}(\tilde{k}, z) = \sum_{j=0}^{\infty} \left[\frac{(-1)^j \tilde{k}^j}{(2j+3)!} \left\{ \left(\frac{Z}{2} \right)^{2j+2} - (2j+3) z^{2j+2} \right\} \right]$$

$$S'_{\text{even}}(\tilde{k}, z) = \sum_{j=0}^{\infty} \left[\frac{(-1)^{j+1} (2j+2) \tilde{k}^j}{(2j+2)!} z^{2j+1} \right]$$

Note that $S'_{\text{odd}}(\tilde{k}, -\frac{Z}{2}) = S'_{\text{odd}}(\tilde{k}, \frac{Z}{2})$ and $S'_{\text{even}}(\tilde{k}, -\frac{Z}{2}) = -S'_{\text{even}}(\tilde{k}, \frac{Z}{2})$. We can rewrite the above equation as

$$2S'_{\text{odd}}(\tilde{k}, \frac{Z}{2}) B^{\xi}_{\text{odd}} = t^{\xi}_1 + t^{\xi}_2 \tag{4.11a}$$

$$2S'_{\text{even}}(\tilde{k}, \frac{Z}{2}) B^{\xi}_{\text{even}} = t^{\xi}_2 - t^{\xi}_1 \tag{4.11b}$$

As long as $S'_{\text{odd}}(\tilde{k}, \frac{Z}{2})$ and $S'_{\text{even}}(\tilde{k}, \frac{Z}{2})$ are both nonzero we have the following solutions

$$B^{\xi}_{\text{odd}} = (t^{\xi}_1 + t^{\xi}_2) \left[\sum_{j=0}^{\infty} \frac{2(2j+2)(-1)^{j+1} \tilde{k}^j}{(2j+3)!} \left(\frac{Z}{2} \right)^{2j+2} \right]^{-1}$$

$$B^{\xi}_{\text{even}} = (t^{\xi}_2 - t^{\xi}_1) \left[\sum_{j=0}^{\infty} \frac{2(-1)^{j+1} \tilde{k}^j}{(2j+1)!} \left(\frac{Z}{2} \right)^{2j+1} \right]^{-1}$$

In order to simplify the above expressions we write the displacement function

$\xi(z)$ in a slightly modified notation as follows

$$\xi(z) = (t_1^\xi + t_1^\xi)\bar{B}_{\text{odd}} S_{\text{odd}}(\tilde{k}, z) + (t_2^\xi - t_1^\xi)\bar{B}_{\text{even}} S_{\text{even}}(\tilde{k}, z)$$

and reformulate the series with $W = \dfrac{\tilde{k}Z^2}{4}$.

$$\bar{B}_{\text{odd}} = \frac{1}{Z^2}\left[\sum_{j=0}^{\infty} \frac{(j+1)(-1)^{j+1}}{(2j+3)!} W^j\right]^{-1}$$

$$\bar{B}_{\text{even}} = \frac{1}{Z}\left[\sum_{j=0}^{\infty} \frac{(-1)^{j+1}}{(2j+1)!} W^j\right]^{-1}$$

The two series above, if expanded in reciprocal, are in fact Laurent series [KNOPP, 1996] and not Taylor series like S_{odd} and S_{even}. The series $\bar{B}_{\text{odd}}$ converges to one of the following expressions depending on the sign of W or in fact that of $\tilde{k}$

$$\bar{B}_{\text{odd}} = \frac{Zk^3}{2kZ\cos\frac{kZ}{2} - 4\sin\frac{kZ}{2}} \quad \text{or} \quad -\frac{Zk^3}{2kZ\cosh\frac{kZ}{2} - 4\sinh\frac{kZ}{2}}$$

and the series $\bar{B}_{\text{even}}$ converges to one of the following expressions

$$\bar{B}_{\text{even}} = -\frac{k}{2\sin\frac{kZ}{2}} \quad \text{or} \quad -\frac{k}{2\sinh\frac{kZ}{2}}$$

It is easy to see that $S'_{\text{even}}(\tilde{k}, \frac{Z}{2})$ is zero for $\tilde{k} = \frac{4\pi^2}{Z^2}$, hence, $\bar{B}_{\text{even}}$ is bounded for $W \in (-\infty, \pi^2)$. In the same domain $\bar{B}_{\text{odd}}$ is also bounded, the first singularity occuring at W being square of the first positive root of $\tan\chi = \chi$, which is larger than $W = \pi^2$. Thus we shall restrict the domain of our solutions for $\tilde{k}$ and hence P accordingly, i.e. $W \in (-\infty, \pi^2)$. A solution with $W > \pi^2$ represents a higher bending mode, with more than two extreme points. These modes do not occur in beams with small deflections, and are, therefore, excluded from our model. Infinitely many solutions of (4.11) exist at $W = \pi^2$ if $t_1^\xi = t_2^\xi$ for $\xi = x, y$ and represents the critical load needed for Euler-buckling [BAŽANT & CEDOLIN, 1991]. One is then required to choose the right solution B_{even}^ξ that conforms to the natural length of the beam (see section 4.5 on page 39). We denote this case as the irregular case.

In order to compute the deformed length and the energy due to bending of the beam we can now integrate $\xi'(z)^2$ and $\xi''(z)^2$ for both $\xi = x$ and y. Here,

the $'$ indicates partial derivatives w.r.t. z. Writing the expressions for $S'_{\text{odd|even}}$ and $S''_{\text{odd|even}}$ in terms of $W = \frac{\tilde{k}Z^2}{4}$ and $w = \tilde{k}z^2$ we have:

$$\begin{aligned}
S'_{\text{odd}}(w) &= \frac{1}{\tilde{k}}\left[\sum_{j=0}^{\infty}\frac{(-1)^{j+1}}{(2j+2)!}w^{j+1} + \sum_{j=0}^{\infty}\frac{(-1)^{j}}{(2j+3)!}W^{j+1}\right] \\
S'_{\text{even}}(w) &= z\sum_{j=0}^{\infty}\frac{(-1)^{j+1}}{(2j+1)!}w^{j} \quad = \quad S''_{\text{odd}}(w) \\
S''_{\text{even}}(w) &= \sum_{j=0}^{\infty}\frac{(-1)^{j+1}}{(2j)!}w^{j}
\end{aligned}$$

The squares of these series can be computed using the rules for products of power series in one variable as follows:

$$\begin{aligned}
S'^2_{\text{odd}}(w) = \frac{1}{\tilde{k}^2}\Bigg[& \sum_{j=0}^{\infty}(-1)^j w^{j+2}\sum_{i=0}^{j}\frac{1}{(2i+2)![2(j-i)+2]!} \\
& + \sum_{j=0}^{\infty}(-1)^j W^{j+2}\sum_{i=0}^{j}\frac{1}{(2i+3)![2(j-i)+3]!} \\
& + 2\sum_{i=0}^{\infty}\frac{(-1)^i W^{i+1}}{(2i+3)!}\sum_{j=0}^{\infty}\frac{(-1)^{j+1}w^{j+1}}{(2j+2)!}\Bigg]
\end{aligned}$$

$$S'^2_{\text{even}}(w) = S''^2_{\text{odd}}(w) = \frac{1}{\tilde{k}}\sum_{j=0}^{\infty}(-1)^j w^{j+1}\sum_{i=0}^{j}\frac{1}{(2i+1)![2(j-i)+1]!}$$

$$S''^2_{\text{even}}(w) = \sum_{j=0}^{\infty}(-1)^j w^{j}\sum_{i=0}^{j}\frac{1}{(2i)![2(j-i)]!}$$

$$\bar{B}^2_{\text{odd}} = \frac{1}{Z^4}\left[\sum_{j=0}^{\infty}(-1)^j b_j^{(1)} W^j\right]^{-1}$$

$$\bar{B}^2_{\text{even}} = \frac{1}{Z^2}\left[\sum_{j=0}^{\infty}(-1)^j b_j^{(2)} W^j\right]^{-1}$$

with

$$b_j^{(1)} = \sum_{i=0}^{j}\frac{(i+1)[(j-i)+1]}{(2i+3)![2(j-i)+3]!} = \frac{2^{2j+1}(j+1)}{(j+3)(2j+4)!}$$

$$b_j^{(2)} = \sum_{i=0}^{j}\frac{1}{(2i+1)![2(j-i)+1]!} = \frac{2^{2j+1}}{(2j+2)!}$$

Due to the odd and even nature of the terms in $\xi(z)$ we can separately integrate $\bar{B}^2_{\text{odd|even}} S'^2_{\text{odd|even}}$ and $\bar{B}^2_{\text{odd|even}} S''^2_{\text{odd|even}}$ for $z \in [-\frac{Z}{2}, \frac{Z}{2}]$ independent of the direction ξ in order to get the expressions for deformed length and bending energy of the beam. Each of the separate integrands being an even function, we integrate twice on the interval $[0, Z/2]$ and change the variable of integration to w. This reformulation yields:

$$2\int_0^{Z/2} S'^2_{\text{odd}}(z)\mathrm{d}z = \int_0^W \frac{S'^2_{\text{odd}}(w)}{\sqrt{\tilde{k}w}}\mathrm{d}w = \frac{Z^5}{32}\sum_{j=0}^{\infty}(-1)^j C_j^{(1)} W^j$$

$$2\int_0^{Z/2} S'^2_{\text{even}}(z)\mathrm{d}z = 2\int_0^{Z/2} S''^2_{\text{odd}}(z)\mathrm{d}z$$
$$= \int_0^W \frac{S'^2_{\text{even}}(w)}{\sqrt{\tilde{k}w}}\mathrm{d}w = \int_0^W \frac{S''^2_{\text{odd}}(w)}{\sqrt{\tilde{k}w}}\mathrm{d}w = \frac{Z^3}{8}\sum_{j=0}^{\infty}(-1)^j C_j^{(2)} W^j$$

$$2\int_0^{Z/2} S''^2_{\text{even}}(z)\mathrm{d}z = \int_0^W \frac{S''^2_{\text{even}}(w)}{\sqrt{\tilde{k}w}}\mathrm{d}w = \frac{Z}{2}\sum_{j=0}^{\infty}(-1)^j C_j^{(3)} W^j$$

with

$$C_j^{(1)} = \sum_{i=0}^{j} \frac{(l+\frac{3}{2})(2i+3)(2l+3) + 2(l+\frac{3}{2})(j+\frac{5}{2}) - 2(2l+3)(j+\frac{5}{2})}{(2l+3)!(2i+3)!(l+\frac{3}{2})(j+\frac{5}{2})}$$
$$= \frac{2^{2j+5}(j+1)}{(2j+6)!} \qquad \text{where } l \equiv j-i$$

$$C_j^{(2)} = \sum_{i=0}^{j} \frac{1}{(2i+1)!(2l+1)!(j+\frac{3}{2})} = \frac{2^{2j+2}}{(2j+3)!}$$

and

$$C_j^{(3)} = \sum_{i=0}^{j} \frac{1}{(2i)!(2l)!(j+\frac{1}{2})} = \frac{2^{2j}}{(2j+1)!}$$

Thus, we get the direction independent odd and even terms in the expressions for the curved length and the bending energy that are expressed as quotients of convergent power series expansions.

$$\widetilde{L}_{\text{odd}} = \frac{1}{32}\sum_{j=0}^{\infty}(-1)^j C_j^{(1)} W^j \left[\sum_{j=0}^{\infty}(-1)^j b_j^{(1)} W^j\right]^{-1}$$

$$\widetilde{L}_{\text{even}} = \frac{1}{8}\sum_{j=0}^{\infty}(-1)^j C_j^{(2)} W^j \left[\sum_{j=0}^{\infty}(-1)^j b_j^{(2)} W^j\right]^{-1}$$

$$\mathcal{E}_{b\text{odd}} = \frac{1}{8}\sum_{j=0}^{\infty}(-1)^j C_j^{(2)} W^j \left[\sum_{j=0}^{\infty}(-1)^j b_j^{(1)} W^j\right]^{-1}$$

$$\mathcal{E}_{b\text{even}} = \frac{1}{2}\sum_{j=0}^{\infty}(-1)^j C_j^{(3)} W^j \left[\sum_{j=0}^{\infty}(-1)^j b_j^{(2)} W^j\right]^{-1}$$

These series quotients, other than where the denominator converges to zero, can be shown to converge to the following expressions.

$$\begin{aligned}\widetilde{L}_{\text{odd}} =& \frac{1}{8}\left[\frac{u^2 - 2\sin^2 u + u\sin u\cos u}{(u\cos u - \sin u)^2}\right] \\ &\text{or} \quad \frac{1}{8}\left[\frac{u^2 - 2\sinh^2 u + u\sinh u\cosh u}{(u\cosh u - \sinh u)^2}\right]\end{aligned} \tag{4.12a}$$

$$\widetilde{L}_{\text{even}} = \frac{1}{8}\left[\frac{2u - \sin 2u}{u(1-\cos 2u)}\right] \quad \text{or} \quad \frac{1}{8}\left[\frac{\sinh 2u - 2u}{u(\cosh 2u - 1)}\right] \tag{4.12b}$$

$$\mathcal{E}_{b\text{odd}} = \frac{1}{4}\left[\frac{u^3(2u - \sin 2u)}{(u\cos u - \sin u)^2}\right] \quad \text{or} \quad \frac{1}{4}\left[\frac{u^3(\sinh 2u - 2u)}{(u\cosh u - \sinh u)^2}\right] \tag{4.12c}$$

$$\mathcal{E}_{b\text{even}} = \frac{1}{2}\left[\frac{u\sin 2u}{1-\cos 2u}\right] \quad \text{or} \quad \frac{1}{2}\left[\frac{u\sinh 2u}{\cosh 2u - 1}\right] \tag{4.12d}$$

where $u^2 = W$, $W > 0$, for the first set of expressions and $u^2 = -W$, $W < 0$, for the second set, thus also $u = \frac{kZ}{2}$. The denominator vanishes for the first time at $W = \pi^2$ as noted earlier, having been contributed by $\bar{B}_{\text{odd|even}}$. These factors, along with the boundary coefficients, in each direction $\xi = x$ and y form the total deformed length and total bending energy as

$$\widetilde{L} = Z + \frac{Z}{2}\sum_{\xi=x,y}[(t_1^\xi + t_2^\xi)^2\widetilde{L}_{\text{odd}} + (t_2^\xi - t_1^\xi)^2\widetilde{L}_{\text{even}}] \tag{4.13}$$

$$\mathcal{E}_b = \frac{EI}{Z}\sum_{\xi=x,y}[(t_1^\xi + t_2^\xi)^2\mathcal{E}_{b\text{odd}} + (t_2^\xi - t_1^\xi)^2\mathcal{E}_{b\text{even}}] \tag{4.14}$$

Here, the $\sum_{\xi=x,y}$ denotes the sum of the terms for both directions x and y because the slopes t_1^ξ, t_2^ξ depend on the particular direction $\xi = x$ or y. As is easy to see, both $\widetilde{L}$ and $\mathcal{E}_b$ are functions of W, in turn of $\tilde{k}$ and consequently of the compressional force P. The energy due to change in length for the compressional force is given by

$$\mathcal{E}_l = -\frac{P^2 L}{2EA} - P(\widetilde{L} - L) \tag{4.15}$$

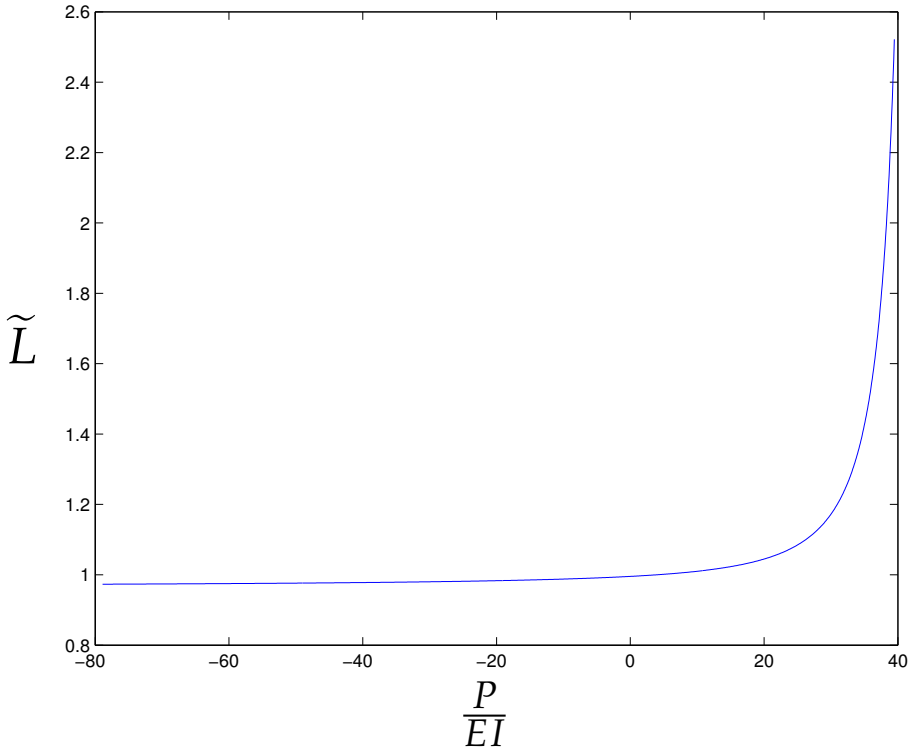


Figure 4.2: Variation of the deformed length

In Figure 4.2 we plot $\widetilde{L}$ as a function of P for a particular example.

Proposition 4.1. *$\widetilde{L}_{odd\,|\,even}$ defined as quotients of series in section 4.4 on the region $-\infty < W < \pi^2$ are positive, monotonically increasing in W, convex and along with their first two derivatives, asymptotically converge to zero as $W \longrightarrow -\infty$.*

The proof is discussed in Appendix A.

4.5 Hooke's law and elastic energy

In order that the state of the beam also has a physical meaning the Hooke's law (4.6) must be satisfied and a unique value P must be determined, which will define the unique state of the beam. This unique state is in fact the maximum of the compression-tension energy with respect to the unknown parameter P. $\mathcal{E}_l$ defined in (4.15) is the Helmholtz free energy [LANDAU & LIFSCHITZ, 1986] due to compression or tension obtained by applying a Legendre transformation [ARNOLD, 1989; ROCKAFELLAR, 1997] to the form in (4.2). The unique P is thus determined as

$$P = \arg\max_{P} \mathcal{E}_l \tag{4.16}$$

In order to write the necessary optimality condition for the above maximisation problem, we require the derivative of this energy w.r.t. P to be zero.

$$\frac{\mathrm{d}\mathcal{E}_l}{\mathrm{d}P} = -\frac{PL}{EA} - \widetilde{L} + L - P\frac{\mathrm{d}\widetilde{L}}{\mathrm{d}P} = 0$$

We may solve (4.16) under the conditions that P is the only unknown variable and Z is already specified via the coordinates of the end-pieces and is constant. We rewrite the condition for optimality of (4.16) in terms of W as

$$-W\frac{\partial\widetilde{L}}{\partial W} - \widetilde{L} - \frac{4ILW}{AZ^2} + L = 0 \tag{4.17}$$

with

$$\frac{\partial\widetilde{L}}{\partial W} = \frac{Z}{2}\sum_{\xi=x,y}[(t_1^\xi + t_2^\xi)^2\frac{\partial\widetilde{L}_{\text{odd}}}{\partial W} + (t_2^\xi - t_1^\xi)^2\frac{\partial\widetilde{L}_{\text{even}}}{\partial W}] \tag{4.18}$$

And for the sake of completeness

$$\frac{\partial E_b}{\partial W} = \frac{EI}{Z}\sum_{\xi=x,y}[(t_1^\xi + t_2^\xi)^2\frac{\partial\mathcal{E}_{b\text{odd}}}{\partial W} + (t_2^\xi - t_1^\xi)^2\frac{\partial\mathcal{E}_{b\text{even}}}{\partial W}] \tag{4.19}$$

For $P = 0$ ($\Rightarrow W = 0$) in (4.17) we also get a relationship between the natural undeformed length L of the beam, the distance between its two endpoints Z and the slopes or orientations of the end pieces expressed as t_1^ξ, t_2^ξ, with $(\xi = x, y)$ as follows:

$$L = Z\left\{1 + \frac{1}{40}[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2] + \frac{1}{12}[(t_2^x - t_1^x)^2 + (t_2^y - t^y)^2]\right\}$$

The above relation confirms the physical observation that changing the orientation of the end pieces to anything nontrivial without exerting a tensile or compressive force will cause the endpoints to come closer to each other; thereby allowing for a deformed shape that is a cubic polynomial along the z-axis, whose length is exactly the same as the natural length of the beam.

Under the assumption of small bending deformation, i.e. extension or compression dominates the total energy, which would happen in the case of extension or if the tangents at both ends were small, the above optimality condition is a modified form of the same condition as the Hooke's law in (4.6). We can thus solve for the unique value of P by using the expressions for $\widetilde{L}$. One may in fact ignore the first term for the case of small bending when solving for the

optimal P.

$$\mathcal{F}(P) = L - \widetilde{L}(P) - \frac{PL}{EA} = 0 \tag{4.20}$$

In case of nontrivial bending deformation, however, we require to solve (4.17) exactly. The derivatives of the odd and even parts of the deformed length $\widetilde{L}$ are:

$$\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W} = \frac{1}{32}\left[\frac{\sum_{j=0}^{\infty}(-1)^j \widetilde{C}_j^{(0)} W^j}{\sum_{j=0}^{\infty}(-1)^j \widetilde{b}_j^{(1)} W^j} - \frac{\sum_{j=0}^{\infty}(-1)^j (j+1) C_{j+1}^{(1)} W^j}{\sum_{j=0}^{\infty}(-1)^j b_j^{(1)} W^j}\right] \tag{4.21a}$$

$$\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W} = \frac{1}{8}\left[\frac{\sum_{j=0}^{\infty}(-1)^j \widetilde{C}_j^{(1)} W^j}{\sum_{j=0}^{\infty}(-1)^j \widetilde{b}_j^{(2)} W^j} - \frac{\sum_{j=0}^{\infty}(-1)^j (j+1) C_{j+1}^{(2)} W^j}{\sum_{j=0}^{\infty}(-1)^j b_j^{(2)} W^j}\right] \tag{4.21b}$$

where

$$\widetilde{C}_j^{(0)} = \frac{2^{2j+9}}{(2j+14)!}[2^{2j+8}(4k^4 + 52k^3 + 223k^2 + 118) + (4k^5 + 104k^4 + 1059k^3 + 5258k^2 + 12677k + 11834)]$$

$$\widetilde{C}_j^{(1)} = \frac{2^{2j+5}}{(2j+7)!}[2^{2j+4}(2j+3) - (2j^2 + 11j + 13)]$$

$$\widetilde{C}_j^{(2)} = \frac{2^{2j+6}}{(2j+11)!}[2^{2j+6}(4x^3 + 28x^2 + 59x + 47) - (4x^4 + 60x^3 + 327x^2 + 777x + 698)]$$

$$\widetilde{b}_j^{(1)} = \frac{2^{2j+4}}{(j+6)(2j+10)!}[2^{2j+6}(2j^3 + 15j^2 + 31j + 7) + (2j^3 + 27j^2 + 115j + 154)]$$

$$\widetilde{b}_j^{(2)} = \frac{2^{2j+3}}{(2j+4)!}(2^{2j+2} - 1)$$

Analogously, one may write the derivatives of odd and even terms of the bending energy expression

$$\frac{\partial \mathcal{E}_{b_{\text{odd}}}}{\partial W} = \frac{1}{8}\left[\frac{\sum_{j=0}^{\infty}(-1)^j \widetilde{C}_j^{(2)} W^j}{\sum_{j=0}^{\infty}(-1)^j \widetilde{b}_j^{(1)} W^j} - \frac{\sum_{j=0}^{\infty}(-1)^j (j+1) C_{j+1}^{(2)} W_j}{\sum_{j=0}^{\infty}(-1)^j b_j^{(1)} W^j}\right] \tag{4.22a}$$

$$\frac{\partial \mathcal{E}_{b_{\text{even}}}}{\partial W} = \frac{1}{2}\left[\frac{\sum_{j=0}^{\infty}(-1)^j \widetilde{C}_j^{(3)} W^j}{\sum_{j=0}^{\infty}(-1)^j \widetilde{b}_j^{(2)} W^j} - \frac{\sum_{j=0}^{\infty}(-1)^j (j+1) C_{j+1}^{(3)} W_j}{\sum_{j=0}^{\infty}(-1)^j b_j^{(2)} W^j}\right] \tag{4.22b}$$

with

$$\widetilde{C}_j^{(3)} = \frac{2^{2j+3}}{(2j+5)!}[2^{2j+3}j + 2^{2j+2} + 1]$$

By substituting the above series expansions in (4.17) we have to solve

$$\begin{aligned}\widetilde{\mathcal{F}}(W) =&[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\left\{-\frac{WZ}{2}\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W} - \frac{Z}{2}\widetilde{L}_{\text{odd}}\right\}\\ &+ [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\left\{-\frac{WZ}{2}\frac{\partial \widetilde{L}_{\text{even}}}{\partial W} - \frac{Z}{2}\widetilde{L}_{\text{even}}\right\}\\ &- \frac{4ILW}{AZ^2} - Z + L\\ =&\,0\end{aligned} \tag{4.23}$$

In case $t_1^\xi = t_2^\xi$ for both $\xi = x, y$ in (4.11) and the solution method suggests going to $W = \pi^2$, i.e. $\lim\limits_{W \nearrow \pi^2} \widetilde{F}(W) \geqslant 0$, we encounter the irregular case as mentioned on page 32. Any arbiterary value for B_{even}^ξ is a solution for (4.11). In this case we may choose any constant β_{even} and replace

$$\beta_{\text{even}} = Z^2 \bar{B}_{\text{even}}^2[(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]$$

in all the above expressions that will satisfy $\widetilde{\mathcal{F}}(\pi^2) = 0$ or $\mathcal{F}(\frac{4EI\pi^2}{Z^2}) = 0$ or $\widehat{\mathcal{F}}(\frac{4EI\pi^2}{Z^2}) = 0$. With such a β_{even} the expressions for deformed length and bending energy from (4.13) and (4.14) reduce to

$$\widetilde{L} = Z + \frac{Z}{2}\left\{[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2\widetilde{L}_{\text{odd}} + \beta_{\text{even}}\frac{1}{8}\sum_{j=0}^{\infty}(-1)^j C_j^{(2)} W^j\right\} \tag{4.24a}$$

$$\mathcal{E}_b = \frac{EI}{Z}\left\{[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\mathcal{E}_{b\text{odd}} + \beta_{\text{even}}\frac{1}{2}\sum_{j=0}^{\infty}(-1)^j C_j^{(3)} W^j\right\} \tag{4.24b}$$

Setting $\widetilde{F}(\pi^2) = 0$ we get

$$\beta_{\text{even}} = -16\frac{\left[1 + \frac{4IL\pi^2}{AZ^3} - \frac{L}{Z} + \frac{1}{2}[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\left\{\pi^2 \left.\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W}\right|_{\pi^2} + \left.\widetilde{L}_{\text{odd}}\right|_{\pi^2}\right\}\right]}{\left[\sum_{j=0}^{\infty}(-1)^j(j+1)C_j^{(2)}\pi^{2j}\right]} \tag{4.25}$$

This solution represents the phenomenon of Euler-buckling [BAŽANT & CE-

DOLIN, 1991] as it adds an arbitrary multiple of the even shape function to the overall deformed shape to attain a maximum of the energy due to compression w.r.t. P at $W = \pi^2$.

Proposition 4.2. *The following two statements hold:*

a. *The root $W_* < \pi^2$ of the equation $\widetilde{\mathcal{F}}(W) = 0$ is unique.*
b. *W_* is smooth w.r.t. the local state variables except in the irregular case, where it is Lipschitz continuous.*

Proof. Proposition 4.1 implies that $\widetilde{F}(W)$ is monotonic (decreasing) and asymptotically linear as $W \to -\infty$. (Also note that $\frac{\partial \widetilde{F}}{\partial W}$ as computed at the end of section 7.3 is always negative, asymptotically going to $-\frac{4IL}{AZ^2}$ as $W \to -\infty$.) We also know from the discussion in section 4.4 on page 32 and section 4.5 on page 39 that $\widetilde{F}(W)$ has a negative pole at $W \to \pi^2$, as $\widetilde{L}_{\text{even}}$ and $\frac{\partial \widetilde{L}_{\text{even}}}{\partial W}$ diverge, and thus, $\widetilde{F}(W)$ has a unique root $W_* < \pi^2$.

In case $t_2^\xi - t_1^\xi = 0$ for both $\xi = x, y$, i.e. the irregular case, $\lim_{W \nearrow \pi^2} \widetilde{F}(W)$ is finite since $\sum_{\xi=x,y}(t_2^\xi - t_1^\xi)^2 = 0$ is the coefficient of the diverging term. In this case if this limit is non-negative we uniquely choose $W_* = \pi^2$ with β_{even} as in (4.25) in order to compute $\widetilde{L}$ and $\mathcal{E}_{b|l}$.

Whenever $W_* < \pi^2$, it is smooth w.r.t. the local state by the implicit function theorem since $\widetilde{L}$ and $\frac{\partial \widetilde{L}}{\partial W}$ and thus $\widetilde{F}$ are smooth w.r.t the local state at each W and $\frac{\partial \widetilde{F}}{\partial W} \neq 0$. The first derivative is given in (7.4) and the second derivative can be obtained by differentiating (4.17) twice w.r.t. $\mathcal{P}_{\text{loc}}$ analogous to (7.4).

If $W_* < \pi^2$ is the root of $\widetilde{F}(W)$ for the local state $\mathcal{P}_{\text{loc}}$ but if $W_*^\varepsilon = \pi^2$ for some neighbouring irregular local state $\mathcal{P}_{\text{loc}} + \boldsymbol{\varepsilon}$ then there must exist some $0 < \delta \leqslant 1$ where $W_*^{\delta\varepsilon} < \pi^2$ for the local state $\mathcal{P}_{\text{loc}} + \underline{\delta}\boldsymbol{\varepsilon}$ for all $0 < \underline{\delta} < \delta$ but $W_*^{\delta\varepsilon} = \pi^2$. Thus at δ we replace the implicit function $W_* = \arg\{\widetilde{F}(W) = 0\}$ by $W_* = \min(\pi^2, \arg\{\widetilde{F}(W) = 0\})$. One may visualise this as if at δ along the ray we have the intersection of a smooth function with a horizontal hyperplane that forms an upper bound for this function. Due to this intersection W_* is at least Lipschitz whenever the irregular case $W_* = \pi^2$ occurs. □

One may try to get away from the singularity in $\mathcal{F}(P)$ by multiplying (4.20) with the common series in the denominator of the expression of $\widetilde{L}$ and evaluate the expression as close to the singular point as needed. We have

$$\begin{aligned}\widehat{\mathcal{F}}(P) &= \left(L - \widetilde{L}(P) - \frac{PL}{EA}\right) \sum_{j=0}^{\infty} (-1)^j \frac{2^{2j+5}}{(2j+8)!} \left[2^{2j+3}(2j^2 + 7j + 4) + j + 3\right] W^j \\ &= 0\end{aligned}$$

with $W = \dfrac{PZ^2}{4EI}$, which can be further simplified by substituting the expression for $\widetilde{L}$ as follows:

$$\begin{aligned}\widehat{\mathcal{F}}(P) = \left(L - Z - \frac{PL}{EA}\right) \sum_{j=0}^{\infty} \widehat{C}_j^{(3)} W^j \\ -\frac{Z}{16}\Bigg[\frac{1}{4}[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2] \sum_{j=0}^{\infty} \widehat{C}_j^{(1)} W^j \\ + [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2] \sum_{j=0}^{\infty} \widehat{C}_j^{(2)} W^j\Bigg] = 0\end{aligned} \tag{4.26}$$

where

$$\widehat{C}_j^{(1)} = (-1)^j \frac{2^{2j+7}}{(2j+8)!}[2^{2j+5} j + j^2 + 7j + 14]$$

$$\widehat{C}_j^{(2)} = (-1)^j \frac{2^{2j+5}}{(2j+9)!}[2^{2j+5}(2j^2 + 9j + 8) - (2j^3 + 19j^2 + 55j + 46)]$$

$$\widehat{C}_j^{(3)} = (-1)^j \frac{2^{2j+5}}{(2j+8)!}\left[2^{2j+3}(2j^2 + 7j + 4) + j + 3\right]$$

The function $\widehat{\mathcal{F}}(P)$ does not have a singularity anymore, however unlike $\mathcal{F}(P)$ it is no longer monotonic w.r.t P.

It is easy to see that if we multiply (4.23) with its common denominator the resulting coefficients are algebraically far too complicated to be evaluated fast and in a numerically stable way. Consequently, we require a Newton iteration with safeguards, which avoids poles as for solving $\mathcal{F}(P) = 0$ or $\widetilde{\mathcal{F}}(W) = 0$.

The safeguarding is necessary on one hand to prevent landing on the singular point or overstepping it, and on the other hand to prevent going very far in the negative region where the series may not have been evaluated precisely. Overstepping the singular point would mean physically having more than two extremal points or more than one points of inflection in the deformed beam. Going too far into the negative region means too much force pulling the beam apart even though the endpoints are fixed at the given boundary conditions. Landing exactly on the singular point we either do not have a unique solution, i.e. the irregular case, or no solution at all to the boundary values specified.

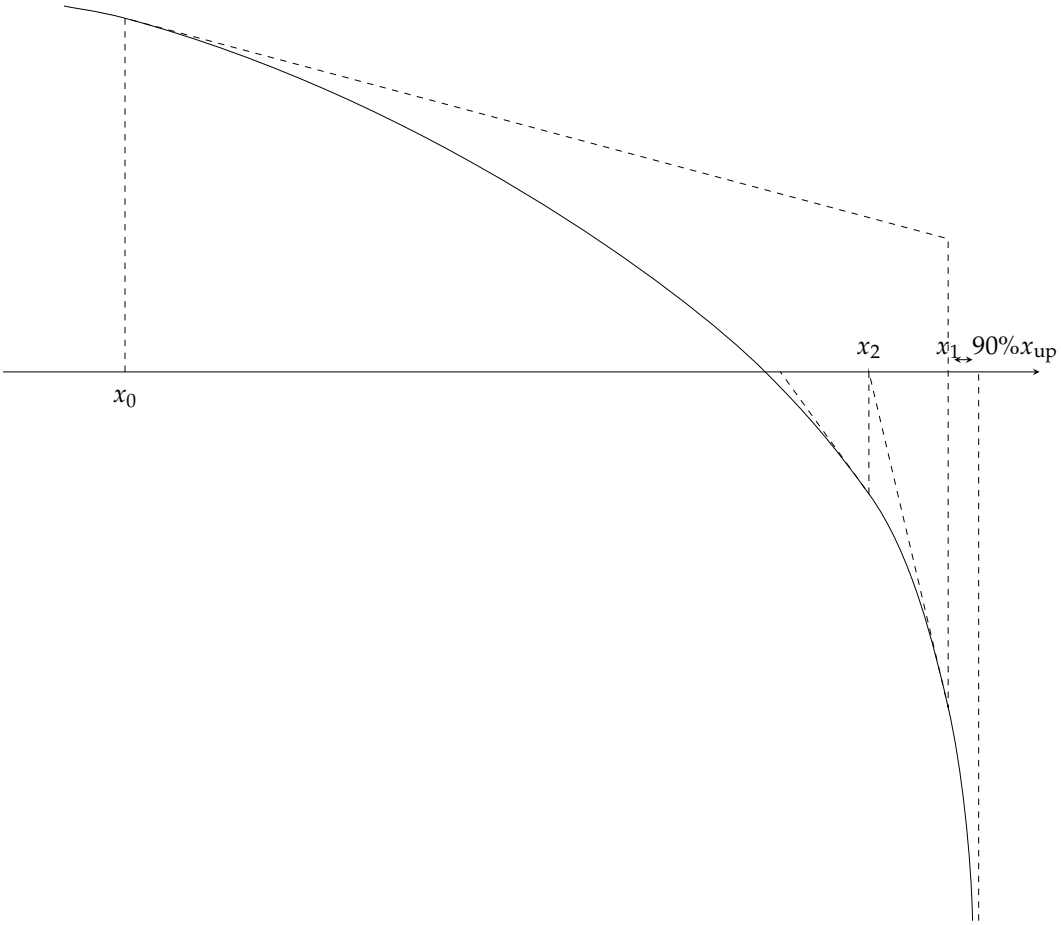


Figure 4.3: Sketch of the safeguarded Newton method

4.6 Safeguarded Newton iteration

In order to use the Newton method to solve (4.26) or (4.23) we require the computation of the derivative $\frac{\mathrm{d}\widetilde{\mathcal{F}}}{\mathrm{d}W}(W)$ at the current value of W in each iteration

$$W_{k+1} = W_k - \widetilde{\mathcal{F}}(W_k) \left(\frac{\mathrm{d}\widetilde{\mathcal{F}}}{\mathrm{d}W}(W_k) \right)^{-1} \tag{4.27}$$

The safeguarding mentioned above is done by constraining the above iteration within an upper and lower bound for W. We first evaluate $\widetilde{\mathcal{F}}(0)$, if the value is negative the deformed length, in this case the cubic length, is smaller than the given natural undeformed length. Thus, we take $W = 0$ as the lower limit and the singular point $W = \pi^2$ as the upper limit. On the other hand if $\widetilde{\mathcal{F}}(0)$ is positive, that the cubic length is bigger than the natural undeformed length, $W = 0$ is taken as the upper limit and we search for a point $W < 0$ where $\widetilde{\mathcal{F}}(W)$

becomes negative, and set that as the lower limit. In each iteration either the lower or the upper limit is updated to the current value of W_k, depending on whether $\widetilde{\mathcal{F}}(W_k)$ at the current iterate W_k is positive or negative. We accept the Newton step if it lies within the 90% distance to the other limit, otherwise we set it to the value which is at 90% distance to the other limiting value and update that limit to the current value. In case the next iterate of the Newton iteration lies completely outside the limiting values and in the wrong direction, i.e. bigger than the current, when the current is also the upper limit, or lesser than the current, when the current is also the lower limit, we bisect the interval and check $\widetilde{\mathcal{F}}(W_{\text{mid}})$ at the mid value, and update the limits accordingly for the next iteration starting with W_{mid} as the current iterate.

The iteration stops when both of the following two conditions are satisfied: the value of $\widetilde{\mathcal{F}}(W_k)$ is small; upon taking a full Newton step, which is acceptable with respect to the two limits but is very small itself, the sign of $\widetilde{\mathcal{F}}(W_{k+1})$ is different from the sign of $\widetilde{\mathcal{F}}(W_k)$. This means that we are in the region close to the intersection point and jumping on either side of it due to roundoff errors.

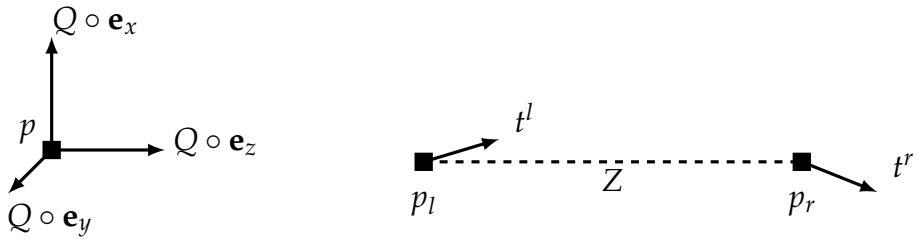
Having found the point that satisfies Hooke's law we can write the expressions for the elastic energy stored in the beam only as a function of the given state, i.e. the given end points and orientations. The sum of both $\mathcal{E}_l$ and $\mathcal{E}_b$ is the total elastic energy. The derivatives of this total energy w.r.t. the end points and orientations can be computed via `ADTAGEO` as long as the last iteration for solving (4.26) or (4.23) performed a full Newton step. This derivative computation constitutes what is called *nested differentiation* as its computation involves solving another system for P where internal derivatives are involved. A full Newton step is necessary at the end to ensure the correctness of outer derivatives.

Chapter 5

Modelling a flexible frame

5.1 Putting beams together

An elastic frame consists of many beams connected together at their endpoints, and maintain rigid angles at the joints with each other under load. The endpoints of the beams, which are also the points of connection with other beams, shall henceforth be called *Frame-points*. An elastic frame can be defined completely, without an external load, by specifying these Frame-points and the beams connecting these, along with the relative orientations of the different beams which connect at one Frame-point. We specify each Frame-point by its position and, to begin with, a neutral orientation, i.e. the quaternion which maps to the unit rotation matrix. This orientation of the Frame-point may change in the course of the optimisation. Each beam is specified by two of the Frame-points as its endpoints and two relative orientations, quaternions, that specify the orientation of the beam at the endpoints w.r.t. the orientation of the Frame-point itself. Each beam also has its own natural length, cross-section and elastic parameters specified. The Frame-points essentially form the joints between two or more beams.



(a) (b)

Figure 5.1: (a) A Frame-point with its internal orientation specified by the quaternion. (b) Specification of the endpoints and tangents of a beam joining two Frame-points.

Let us look at this setup closely. We have a list $\mathcal{P}$ with $\mathcal{P}_i$ being Frame-points. A Frame-point consists of a position vector and a quaternion, i.e. $\mathcal{P}_i = (p_i, Q_i)$. Let $\mathcal{B}$ be a list of beams with each beam described by $\mathcal{B}_j = (l_j, r_j, C_j^l, C_j^r, \mathfrak{D}_j)$. The endpoints of beam $\mathcal{B}_j$ are specified as indices (or pointers) l_j and r_j in the list $\mathcal{P}$, i.e. the position vectors of the endpoints are p_{l_j} and p_{m_j}. The quaternions C_j^l and C_j^r specify relative orientation at the two endpoints. The tangent vectors at the two endpoints may be written as

$$\begin{aligned} t_j^l &= \frac{C_j^l Q_{l_j}}{\|Q_{l_j}\|} \circ \mathbf{e}_z \\ t_j^r &= \frac{C_j^r Q_{r_j}}{\|Q_{r_j}\|} \circ \mathbf{e}_z \end{aligned}$$

Here, we need to normalise the quaternions Q_{l_j} and Q_{r_j} because although we start with quaternions on the unit sphere, they may change in the course of the optimisation procedure. The quaternions C_j^l and C_j^r are constants (see section on Rigid joints), so we do not need to normalise them. Note that the product of two quaternions is not commutative and thus the above products need to be computed carefully. The action of the same product quaternion on $\mathbf{e}_x$ and $\mathbf{e}_y$ provide vectors along the cross-sectional axes of the beam. The information about the cross-section and elastic parameters of the beam are written above as one symbol $\mathfrak{D}_j$ (the design). Thus, we can compute the state of each beam and the elastic energy stored in it. The sum of the elastic energies of individual beams is the total elastic energy stored in the frame.

$$\mathcal{E}_{\text{elastic}} = \sum_{j=0}^{\text{sizeof}(\mathcal{B})} [\mathcal{E}_b(\mathcal{B}_j) + \mathcal{E}_l(\mathcal{B}_j)]$$

with $\mathcal{E}_l$ and $\mathcal{E}_b$ defined in (4.15) and (4.14). The state variables for the frame are thus only the position vectors and quaternions associated with the Frame-points in the list $\mathcal{P}$.

5.2 Rigid joints

For our optimisation purpose we require the joints to be rigid. The joint as a whole rigid body, however, may change its orientation. Therefore, the orientations of all the beams connected at the joint should change simultaneously, but never relative to each other. This is maintained by allowing only the qua-

ternions associated with the Frame-point itself to change during the optimisation, but not the quaternions that specify relative orientations for the endpoints of the beams. So for any Frame-point $\mathcal{P}_i$ its orientation given by Q_i may change, but for any beam $\mathcal{B}_j$ the relative orientations at the endpoints specified by C_j^l and C_j^r are kept always constant. The total relative rotation from beam $\mathcal{B}_j$ to beam $\mathcal{B}_k$ with one common endpoint, say $l_j = l_k$, at the common point is given by the quaternion $C_k^l C_j^{l^*}$. Thus, the angle that the two beams make to each other at the joint is kept constant. The tangent vectors of both the beams at the common point, however, change whenever Q_{l_j} (which is the same as Q_{l_k}) changes. Hence, the joint acts as a rigid joint.

5.3 Masses and external forces

The masses of the beams, external masses attached to the frame and external forces which may act on a frame are modelled as placed at or acting on special points, which we shall call *Support-points*. The masses, whether of the beams themselves, or external masses attached to the frame, exert force on the frame at their particular Support-point in the presence of gravity or acceleration of the entire frame. Each force or mass is associated with a Support-point whose position is variable in the frame. Thus, we have a list of Support-points $\mathfrak{S}$ with $\mathfrak{S}_i = (\mathcal{S}_i, \texttt{S\_func}_i(), m_i|f_i)$. Here, $\mathcal{S}_i$ is a sublist of $\mathcal{P}$, $\texttt{S\_func}_i()$ is a function which takes the sublist $\mathcal{S}_i$ as its argument and returns one position vector, and $m_i|f_i$ is either the mass or the force attached to this Support-point. In case a mass m_i is attached to the Support-point the force vector f_i is computed by multiplying with the gravity field vector or the acceleration vector.

The work done by these forces is then computed as the scalar product of the force vector with the position vector of this Support-point.

$$\mathcal{W}_{\text{support}} = \sum_{i=1}^{\text{sizeof}(\mathfrak{S})} \texttt{S\_func}_i(\mathcal{S}_i) \cdot f_i$$

The function $\texttt{S\_func}_i()$ may be a user supplied function or in special cases compute the centroid of the positions of the Frame-points in $\mathcal{S}_i$. In the case of the mass of the beams themselves the centroid function is used. For the simplicity of implementation the user supplied function can only be some linear combination of the position vectors in $\mathcal{S}_i$ plus some constant vector. In this way we do not add new variables for positions of the Support-points to the state variables as they are dependent on the positions of Frame-points, which

are already state variables. The sublists $\mathcal{S}_i$ of $\mathcal{P}$ are also specified as indices (or pointers) in $\mathcal{P}$.

5.4 Static equilibrium

For a given set of external forces and masses, we can establish the state of static equilibrium by minimising the sum of the total elastic energy and the negative external work. The only constraint is that the quaternions that determine the orientations of the Frame-points must lie on the unit sphere $S(\mathbb{R}^4)$. This constraint is inserted as the following renormalization energy term in the objective

$$\mathfrak{R}_{\text{orient}} = \chi \cdot \sum_{i=1}^{\text{sizeof}(\mathcal{P})} (\|Q_i\| - 1)^2$$

with χ being some constant that provides the correct physical dimensions and scaling. Thus, the minimisation proceeds as an unconstrained minimisation of the energy functional

$$\min_{\mathcal{P}} \mathcal{E}(\mathcal{P}) := \mathcal{E}_{\text{elastic}} - \mathcal{W}_{\text{support}} + \mathfrak{R}_{\text{orient}}$$

Here, the derivatives of the objective with respect to the state variables are available through `ADTAGEO` as discussed before. One may also derive these derivatives from the expressions in chapter 4 using the implicit function theorem (see chapter 7). The minimisation is done using an the total quasi-Newton method discussed in chapter 3.

5.5 Design variables

The elastic parameters, Young's modulus and Poisson's ratio, along with the cross-sectional radii, the natural length of the beam and density for its material together describe the design of the beams in the frame. For each beam they were denoted on page 46 as $\mathfrak{D}_j$. For all the beams in the frame together form the set of design variables. These are treated as constants in the static equilibrium analysis of the structure, but are variable when a design optimisation is performed. The energy of the structure depends not only on the variables $\mathcal{P}$ but also on $\mathfrak{D}$ in the form of crossection area, area moment of inertia, elasticity parameters of the material etc. Thus we write the energy as $\mathcal{E}(\mathfrak{D}, \mathcal{P})$ during

the design optimisation. The design variables define a cost functional $\mathcal{M}(\mathfrak{D})$ directly, independent of the stable state equilibrium, which is a measure of the desirability of a particular design, e.g. low cost designs are preferred over high cost ones. The cost functional, however, is not the only measure for optimality. The energy achieved by static equilibrium is a measure of the stiffness of a structure and in order to have an optimal design we need to minimise cost while maximising stiffness.

Chapter 6

Design optimisation

6.1 Saddle point formulation

For each reasonable design $\mathfrak{D}$ and load scenario there are stable states that form the local minimum of the energy functional $\mathcal{E}(\mathfrak{D}, \mathcal{P})$. A simple design goal is

$$\min_{\mathfrak{D}} \mathcal{M}(\mathfrak{D}) \quad \text{s.t. } \varepsilon = \min_{\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) \quad \text{for some } \varepsilon \tag{6.1}$$

This is a case of bilevel programming [DEMPE, 2002; LOU et al., 1996a], and we first write the KKT conditions for the inner programme.

$$\min_{\mathfrak{D}} \mathcal{M}(\mathfrak{D}) \quad \text{s.t. } \mathcal{E}(\mathfrak{D}, \mathcal{P}) = \varepsilon \text{ and } \nabla_{\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) = 0$$

Then the KKT conditions for this programme along with the equality constraints are the following

$$\begin{array}{lcl} 0 & = & \mu \nabla_{\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) + \lambda^{\top} \nabla^2_{\mathcal{P}\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) = 0 \\ \nabla_{\mathfrak{D}} \mathcal{M}(\mathfrak{D}) & = & \mu \nabla_{\mathfrak{D}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) + \lambda^{\top} \nabla^2_{\mathcal{P}\mathfrak{D}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) = 0 \end{array}$$

In the vicinity of the solution $(\mathfrak{D}^*, \mathcal{P}^*)$ the Hessian of the energy functional must be positive definite and its gradient zero in order for it to be a stable state solution. Thus, we have $\nabla^2_{\mathcal{P}\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) \succ 0$ for all $(\mathfrak{D}, \mathcal{P})$ sufficiently close to $(\mathfrak{D}^*, \mathcal{P}^*)$, along with $\nabla_{\mathcal{P}} \mathcal{E}(\mathfrak{D}^*, \mathcal{P}^*) = 0$ forcing the Lagrange multipliers $\lambda = 0$. This means we have no strict complementarity. The loss of usual constraint qualifications, like linear independence or Mangasarin - Fromowitz constraint qualification, in the KKT reformulation of bilevel programmes is well known, see e.g. SCHEEL & SCHOLTES [2000].

Design optimisations problems like those formulated in BENDSØE & SIGMUND [2003], though often initially bilevel programmes, may be reduced to a

constrained optimisation problem due to the energy being quadratic in terms of the state variables. Such a quadratic energy reduces the lower level optimisation problem to the solution of an equation system. The energy as define in chapter 4 for one beam and in chapter 5 for the total frame is in contrast very highly nonlinear, even nonquadratic, in the state variables due to the terms that are quotients of the series expansions as well as implicit dependence introduced via the solution of the the Hooke's law equation (4.23). One cannot therefore get away from a bilevel programme and must solve a nonlinear saddle point problem.

A Newton iteration with line-search along a suitable merit function may be used to solve for the saddle point. The merit function should be exact in that it attains a minimum at the saddle point of the above system. We drop λ, rescale the KKT equations by $1/\mu$ and compute the corresponding Newton step by solving the following system

$$\frac{1}{\mu}\mathcal{H}(\mathfrak{D},\mathcal{P})\begin{bmatrix}\delta\\ \varsigma\end{bmatrix} = -\begin{bmatrix}\frac{1}{\mu}\mathcal{M}_{\mathfrak{D}}(\mathfrak{D}) - \mathcal{E}_{\mathfrak{D}}(\mathfrak{D},\mathcal{P})\\ -\mathcal{E}_{\mathcal{P}}(\mathfrak{D},\mathcal{P})\end{bmatrix} \tag{6.2}$$

with the Hessian of the Lagrangian $\mathcal{H}(\mathfrak{D},\mathcal{P})$ given by

$$\mathcal{H}(\mathfrak{D},\mathcal{P}) = \mu\begin{bmatrix}\frac{1}{\mu}\nabla^2_{\mathfrak{D}\mathfrak{D}}\mathcal{M}(\mathfrak{D}) - \nabla^2_{\mathfrak{D}\mathfrak{D}}\mathcal{E}(\mathfrak{D},\mathcal{P}) & -\nabla^2_{\mathfrak{D}\mathcal{P}}\mathcal{E}(\mathfrak{D},\mathcal{P})\\ -\nabla^2_{\mathcal{P}\mathfrak{D}}\mathcal{E}(\mathfrak{D},\mathcal{P}) & -\nabla^2_{\mathcal{P}\mathcal{P}}\mathcal{E}(\mathfrak{D},\mathcal{P})\end{bmatrix} \tag{6.3}$$

Due to λ being all zero in the KKT conditions the following function will have exactly the same local minima as the original bilevel problem (6.1)

$$\begin{aligned}\min_{(\mathfrak{D},\mathcal{P})}\mathcal{F}_\mu(\mathfrak{D},\mathcal{P}) := \min_{(\mathfrak{D},\mathcal{P})}&\frac{1}{\mu}\mathcal{M}(\mathfrak{D}) - \mathcal{E}(\mathfrak{D},\mathcal{P})\\ &+\left[\nabla_{\mathcal{P}}\mathcal{E}(\mathfrak{D},\mathcal{P})\right]^\top\left[\nabla^2_{\mathcal{P}\mathcal{P}}\mathcal{E}(\mathfrak{D},\mathcal{P})\right]^{-1}\left[\nabla_{\mathcal{P}}\mathcal{E}(\mathfrak{D},\mathcal{P})\right]\end{aligned} \tag{6.4}$$

The final term in the function $\mathcal{F}_\mu$ is an exact penalty for deviation from stable state where $\nabla_{\mathcal{P}}\mathcal{E}(\mathfrak{D},\mathcal{P}) \neq 0$. Hence, $\mathcal{F}_\mu$ can be used as an exact merit function for testing descent of the Newton direction (δ,ς) [LOU et al., 1996b].

The directional derivative of the merit function $\mathcal{F}_\mu$ in the Newton direction can be approximated by multiplying with the row vector (δ,ς) from the left in

equation (6.2).

$$\begin{aligned}
&\frac{1}{\mu}\delta^\top \nabla_{\mathfrak{D}} \mathcal{M}(\mathfrak{D}) - \delta^\top \nabla_{\mathfrak{D}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) + \varsigma^\top \nabla_{\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) \\
&\quad = -\delta^\top \big(\frac{1}{\mu}\nabla^2_{\mathfrak{D}\mathfrak{D}} \mathcal{M}(\mathfrak{D}) - \nabla^2_{\mathfrak{D}\mathfrak{D}} \mathcal{E}(\mathfrak{D}, \mathcal{P})\big)\delta - \varsigma^\top \nabla^2_{\mathcal{P}\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P})\varsigma \\
&\quad = \nabla \mathcal{F}_\mu^\top \begin{bmatrix} \delta \\ \varsigma \end{bmatrix} + \mathrm{O}(\|\delta\|\|\nabla_{\mathcal{P}}\mathcal{E}\| + \|\delta\|\|\nabla_{\mathcal{P}}\mathcal{E}\|^2 + \|\varsigma\|\|\nabla_{\mathcal{P}}\mathcal{E}\|^2)
\end{aligned}$$

Thus, that we have descent, i.e.

$$\nabla \mathcal{F}_\mu^\top \begin{bmatrix} \delta \\ \varsigma \end{bmatrix} < 0$$

is guaranteed upto higher order terms under the concavity-convexity condition

$$\nabla^2_{\mathfrak{D}\mathfrak{D}} \mathcal{M}(\mathfrak{D}) - \nabla^2_{\mathfrak{D}\mathfrak{D}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) \succ 0\,, \qquad \nabla^2_{\mathcal{P}\mathcal{P}} \mathcal{E}(\mathfrak{D}, \mathcal{P}) \succ 0 \tag{6.5}$$

This is also the second-order sufficiency condition for $(\mathfrak{D}, \mathcal{P})$ being a saddle point of (6.1).

The Hessian $\mathcal{H}(\mathfrak{D}, \mathcal{P})$ in (6.3) will have a special structure with the condition (6.5). This is a *quasi-definite* Hessian [GILL et al., 1996; VANDERBEI, 1995]. The upper diagonal block is positive definite while the lower diagonal block is negative definite. Such a quasi-definite matrix H can be factorised by a modified Cholesky method in a special $LDL^\top$ form:

$$H = \begin{bmatrix} C & 0 \\ B & G \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} C^\top & B^\top \\ 0 & G^\top \end{bmatrix} \tag{6.6}$$

where C and G are lower triangular and B is a full matrix.

Thus, we can use this structure of the Hessian to implement a Newton algorithm that uses the factorisation from (6.6) to solve for the Newton direction and then perform a line-search on the penalised merit function. We need, however, to compute the Hessian using AD and then factorise it maintaining the structure in (6.6). The factorisation can maintain this structure as long as we take the following precaution. Whenever a pivot with the wrong sign is encountered in the factorisation process we replace it by a pivot with the correct sign that dominates its remaining unfactorised row and column.

Unfortunately `ADTAGEO` is, as of now, unable to provide us with second derivatives that are necessary to compute the Hessian (6.3). So we need to rely

on quasi-Newton methods [NOCEDAL & WRIGHT, 1999]. However, no quasi-Newton solver is known for solving saddle point problems.

6.2 Designs as dual variables

A further simplification of the saddle point problem (6.1) is possible by taking a closer look at the involved parameters and functions. If we define the design variables $\mathfrak{D}$ to be the areas of crossections of the beams in the frame, then the functions $\mathcal{E}_{\text{elastic}}(\mathfrak{D}, \mathcal{P})$ and $\mathcal{M}(\mathfrak{D})$ depend only linearly on the components of $\mathfrak{D}$. For each beam in the frame we can define $e_i(\mathcal{P})$, energy density, and ρ_i, mass or cost density, such that

$$\mathcal{E}_{\text{elastic}}(\mathfrak{D}, \mathcal{P}) = \sum_{i=1}^{\text{sizeof}(\mathcal{B})} d_i e_i(\mathcal{P})$$

$$\mathcal{M}(\mathfrak{D}) = \sum_{i=1}^{\text{sizeof}(\mathcal{B})} d_i \rho_i$$

where $\mathfrak{D} = \left(d_i\right)_{i=1}^{\text{sizeof}(\mathcal{B})} \geqslant 0$. In the notation of section 4.4 one can write for beam i, $e_i(\mathcal{P}) = \frac{1}{A}\left(\mathcal{E}_b + \mathcal{E}_l\right)$ with $\mathcal{E}_b$ and $\mathcal{E}_l$ as defined in (4.14) and (4.15) with $d_i = A$. The design optimisation goal (6.1) is then an optimal sizing problem.

6.2.1 Constrained minimisation formulation

Rewriting the design goal in terms of the above notation we have

$$\min_{\mathcal{P}} \max_{\mathfrak{D}} \left[-\mathcal{W}_{\text{support}}(\mathcal{P}) + \sum_{i=1}^{\text{sizeof}(\mathcal{B})} d_i \left(e_i(\mathcal{P}) - \frac{\rho_i}{\mu} \right) \right] \tag{6.7}$$

If viewed as a saddle point problem on a Lagrangian function, the programme (6.7) can be deconstructed into a constrained minimisation problem in only the state variables $\mathcal{P}$

$$\min_{\mathcal{P}} -\mathcal{W}(\mathcal{P}) \quad \text{s. t.} \quad e_i(\mathcal{P}) - \frac{\rho_i}{\mu} \leqslant 0 \forall i = 1 \ldots \text{sizeof}(\mathcal{B}) \tag{6.8}$$

Here, the design variables take the role of Lagrange multipliers or dual variables. The parameter μ takes the role of a compromise parameter between the elastic compliance and mass or cost of the beams, thus, for varying values of μ

we can obtain a Pareto front of solutions.

Writing the first order optimality conditions for (6.8) we have

$$-\nabla_{\mathcal{P}}\mathcal{W}(\mathcal{P}) + \sum_{i=1}^{\text{sizeof}(\mathcal{B})} d_i \nabla_{\mathcal{P}} e_i(\mathcal{P}) = 0 \tag{6.9a}$$

$$d_i \geqslant 0 \qquad \forall\, i = 1 \ldots \text{sizeof}(\mathcal{B}) \tag{6.9b}$$

$$e_i(\mathcal{P}) - \frac{\rho_i}{\mu} \leqslant 0 \qquad \forall\, i = 1 \ldots \text{sizeof}(\mathcal{B})$$

along with the complementarity condition

$$\begin{gathered} d_i \left(e_i(\mathcal{P}) - \frac{\rho_i}{\mu} \right) = 0 \Longrightarrow \\ \left(d_i = 0 \wedge e_i(\mathcal{P}) - \frac{\rho_i}{\mu} < 0 \right) \vee \left(d_i > 0 \wedge e_i(\mathcal{P}) - \frac{\rho_i}{\mu} = 0 \right) \\ \forall\, i = 1 \ldots \text{sizeof}(\mathcal{B}) \end{gathered} \tag{6.10}$$

Notice that (6.9a) is exactly the same condition as the optimality condition for the inner problem in (6.1), i.e. the condition for a stable state. The non-negativity constraint (6.9b) on the Lagrange multipliers enforces that we have physically appropriate design variables as a negative area value is physically senseless. The constraints themselves are a compromised balance between the mass, i.e. the cost of a particular beam, and the elastic energy due to its deformation that represents its compliance to deformation. Enforcing strict complementarity (6.10) gives an indication of the usefulness of having a beam in the frame structure at all. When the mass of a beam is too much compared to its stiffness in the structure the beam may be removed by assigning it a zero valued design; however, when the mass and the stiffness balance each other the beam must have a positive design value. This formulation requires us to use an optimisation method that always follows a dual feasible path fulfilling (6.9b) at each step. Thus, a design optimisation problem is in fact reduced to a constrained state optimisation problem.

At the optimal, all the beams for which the mass density dominates the elastic energy density for that beam upto the compromise parameter will vanish, and only those beams will remain where these two densities balance each other. The optimal structure is one in an energetic equilibrium where the elastic deformation energy, and thus, its elastic compliance is limited to be in a compromising balance with the mass of that structure.

6.2.2 Topological sensitivity

Since the term $e_i(\mathcal{P}) - \rho_i/\mu$ governs the direction in which the design variables d_i adjusts during the optimisation process, it can be construed to be a topological sensitivity. This can also be used to determine the effect of adding an extra beam between two given points on the structure. An extra beam with a zero design value is simulated to be present between the given points and deformed according to the state $\widetilde{\mathcal{P}}$ of the extended structure and the term $e(\widetilde{\mathcal{P}}) - \rho/\mu$ is computed. Here, $\widetilde{\mathcal{P}}$ is an extension of the state $\mathcal{P}$ to include the new beam, with two extra positions and orientations. The new positions and orientations are computed using interpolations of the deformed shape of the exiting beams on which the given points lie.

The sign of $e(\widetilde{\mathcal{P}}) - \rho/\mu$ gives an indication how the addition of an extra beam changes the original structure. A negative sign implies that the feasibility of the new structure is preserved. The means that in order to obtain optimality a zero design value must be maintained. The structure essentially does not need to have a beam in such a location. A positive sign on the other hand implies that the state of the structure becomes infeasible by the addition of the new beam and further optimisation steps are required that would lead to a better optimal design. This would invariably assign the design variable for the new beam a non-zero value to obtain optimality. If the sensitivity turns out to be zero it violates only the strict complementarity condition but the state of the structure remains feasible. Assigning any non-zero design value to the new beam in this case will have the structure satisfy all the conditions for optimality. Thus, having or not having the new beam has no effect on the structure of the frame energetically.

Starting from a base design for any frame one can follow an iterative interactive design process. In each iteration of this design process, since the solution of the optimal design problem (6.8) only resizes or deletes the beams, this topological sensitivity at the stable state for the optimal base design can be used as a decision tool to determine the positions where new beams may be added and then recomputing the optimal design with increased number of beams.

Chapter 7

Derivatives revisited

The computation of derivatives, gradients and Jacobian-vector products as described in chapter 2 ensures a maximal cost of computation to be a small multiple of the cost of the function evaluation. However, for our purposes this may still turn out to be several hours of computation in order to optimise a frame and compute the topological sensitivities. Looking a bit deeper at equation (4.17) one can see that *nested differentiation* is essentially the same as using the implicit function theorem to compute the derivatives. Here, let us reconsider the expressions for the energy again and compute the explicit expressions for their derivatives with respect to the coordinates and orientations of the endpoints. This would save some computation time due to cancellations at the optimal W as well as amortisation of the overhead due to AD.

7.1 From global to local coordinates

Using the unit vector along $p_2 - p_1$ we define a local coordinate system for each beam where $\mathbf{e}_z = \dfrac{p_2 - p_1}{\|p_2 - p_1\|}$. As noted in chapter 4 any two mutually orthogonal unit vectors that form a right handed orthonormal coordinate basis may be chosen to be $\mathbf{e}_x$ and $\mathbf{e}_y$. In order to compute the derivatives we require a concrete choice. From the geometry of $\mathbb{R}^3$ let us consider two simple ways to form a local orthonormal coordinate system by choosing an $\mathbf{e}_x$ and $\mathbf{e}_y$, i.e. using simple vector algebra and using a quaternion.

Using simple vector algebra choose mutually orthogonal unit vectors as $\mathbf{e}_x$ and $\mathbf{e}_y$, such that

$$(\mathbf{e}_x^{x_g}, \mathbf{e}_x^{y_g}, \mathbf{e}_x^{z_g}) = \frac{1}{s_x}(-\mathbf{e}_z^{z_g}, 0, \mathbf{e}_z^{x_g}) \quad \text{with} \quad s_x = \sqrt{\mathbf{e}_z^{x_g 2} + \mathbf{e}_z^{z_g 2}}, \quad \mathbf{e}_y = \mathbf{e}_z \times \mathbf{e}_x$$

This yields Jacobian matrices

$$\frac{\mathrm{d}\mathbf{e}_y}{\mathrm{d}\mathbf{e}_z} = \frac{1}{s_x}\begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} - \frac{1}{s_x^3}\begin{bmatrix} -\mathbf{e}_z^{z_g} \\ 0 \\ \mathbf{e}_z^{x_g} \end{bmatrix}\begin{bmatrix} \mathbf{e}_z^{x_g} & 0 & \mathbf{e}_z^{z_g} \end{bmatrix} \tag{7.1a}$$

$$\frac{\mathrm{d}\mathbf{e}_x}{\mathrm{d}\mathbf{e}_z} = \begin{bmatrix} 0 & \mathbf{e}_z^{x_g} & 0 \\ -\mathbf{e}_z^{x_g} & 0 & -\mathbf{e}_z^{z_g} \\ 0 & \mathbf{e}_z^{z_g} & 0 \end{bmatrix} + \begin{bmatrix} 0 & -\mathbf{e}_z^{z_g} & \mathbf{e}_z^{y_g} \\ \mathbf{e}_z^{z_g} & 0 & -\mathbf{e}_z^{x_g} \\ -\mathbf{e}_z^{y_g} & \mathbf{e}_z^{x_g} & 0 \end{bmatrix}\frac{\mathrm{d}\mathbf{e}_x}{\mathrm{d}\mathbf{e}_z} \tag{7.1b}$$

This choice may have a degeneracy in case both $\mathbf{e}_z^{x_g}$ and $\mathbf{e}_z^{z_g}$ are zero. One may avoid that by looking for the largest and the smallest coordinate components, choosing the negative of the largest component, the smallest component as is and switching the coordinate indices to form $\mathbf{e}_x$. The derivatives are then computed accordingly.

Using a quaternion that rotates the vector $(0,0,1)$ in global coordinates to the unit vector $\mathbf{e}_z$. Such a quaternion is

$$Q_{\text{sys}} = \begin{bmatrix} \sqrt{\dfrac{\mathbf{e}_z^{z_g}+1}{2}}, & -\dfrac{\mathbf{e}_z^{y_g}}{\sqrt{2\mathbf{e}_z^{z_g}+2}}, & \dfrac{\mathbf{e}_z^{x_g}}{\sqrt{2\mathbf{e}_z^{z_g}+2}}, & 0 \end{bmatrix}$$

In general a unit vector $\hat{u}$ is rotated to a unit vector $\hat{v}$ by the quaternion written in terms of the scalar product and the cross product of the two unit vectors as $\begin{bmatrix} \sqrt{\frac{\hat{u}\cdot\hat{v}+1}{2}}, & \frac{\hat{u}\times\hat{v}}{\sqrt{2\hat{u}\cdot\hat{v}+2}} \end{bmatrix}$ with the vector valued cross product containing the three imaginary components of the quaternion. The corresponding rotation matrix is written using (4.1) as

$$Q_{\text{sys}} \mapsto \begin{bmatrix} \dfrac{1+\mathbf{e}_z^{z_g}-\mathbf{e}_z^{x_g{}^2}}{\mathbf{e}_z^{z_g}+1} & -\dfrac{\mathbf{e}_z^{x_g}\mathbf{e}_z^{y_g}}{\mathbf{e}_z^{z_g}+1} & \mathbf{e}_z^{x_g} \\ -\dfrac{\mathbf{e}_z^{x_g}\mathbf{e}_z^{y_g}}{\mathbf{e}_z^{z_g}+1} & \dfrac{1+\mathbf{e}_z^{z_g}-\mathbf{e}_z^{y_g{}^2}}{\mathbf{e}_z^{z_g}+1} & \mathbf{e}_z^{y_g} \\ -\mathbf{e}_z^{x_g} & -\mathbf{e}_z^{y_g} & \mathbf{e}_z^{z_g} \end{bmatrix}$$

The vectors $\mathbf{e}_x$ and $\mathbf{e}_y$ are now obtained by the action of Q_{sys} on the global vectors $(1,0,0)$ and $(0,1,0)$.

$$\mathbf{e}_x^\top = \begin{bmatrix} \frac{1+\mathbf{e}_z^{z_g}-\mathbf{e}_z^{x_g{}^2}}{\mathbf{e}_z^{z_g}+1}, & -\frac{\mathbf{e}_z^{x_g}\mathbf{e}_z^{y_g}}{\mathbf{e}_z^{z_g}+1}, & -\mathbf{e}_z^{x_g} \end{bmatrix};\ \mathbf{e}_y^\top = \begin{bmatrix} -\frac{\mathbf{e}_z^{x_g}\mathbf{e}_z^{y_g}}{\mathbf{e}_z^{z_g}+1}, & \frac{1+\mathbf{e}_z^{z_g}-\mathbf{e}_z^{y_g{}^2}}{\mathbf{e}_z^{z_g}+1}, & -\mathbf{e}_z^{y_g} \end{bmatrix}$$

One writes the derivatives as

$$\frac{\mathrm{d}\mathbf{e}_x}{\mathrm{d}\mathbf{e}_z} = \frac{1}{\mathbf{e}_z^{z_g}+1}\begin{bmatrix} -2\mathbf{e}_z^{x_g} & 0 & 1-\dfrac{1+\mathbf{e}_z^{z_g}-\mathbf{e}_z^{x_g{}^2}}{\mathbf{e}_z^{z_g}+1} \\ -\mathbf{e}_z^{y_g} & -\mathbf{e}_z^{x_g} & \dfrac{\mathbf{e}_z^{x_g}\mathbf{e}_z^{y_g}}{\mathbf{e}_z^{z_g}+1} \\ -1-\mathbf{e}_z^{z_g} & 0 & 0 \end{bmatrix} \tag{7.1c}$$

$$\frac{\mathrm{d}\mathbf{e}_y}{\mathrm{d}\mathbf{e}_z} = \frac{1}{\mathbf{e}_z^{z_g}+1}\begin{bmatrix} -\mathbf{e}_z^{y_g} & -\mathbf{e}_z^{x_g} & \dfrac{\mathbf{e}_z^{x_g}\mathbf{e}_z^{y_g}}{\mathbf{e}_z^{z_g}+1} \\ 0 & -2\mathbf{e}_z^{y_g} & 1-\dfrac{1+\mathbf{e}_z^{z_g}-\mathbf{e}_z^{y_g{}^2}}{\mathbf{e}_z^{z_g}+1} \\ 0 & -1-\mathbf{e}_z^{z_g} & 0 \end{bmatrix} \tag{7.1d}$$

Note that here and in the following the superscripts x_g, y_g, z_g denote coordinates in the global coordinate system of the frame, and the superscript x, y, z denote the coordinate in the local coordinate system of a single beam.

Either one of the above two methods may be used to choose the local coordinate basis in non-degenerate situations, though using a quaternion to rotate avoids degeneracies and numerical inaccuracies [EVANS & MURAD, 1977; GOLDSTEIN, 1973; GRIEWANK et al., 1979].

The distance between the two endpoints p_1 and p_2 denoted in chapter 4 as Z is the key variable in the local coordinate system of a single beam that is used to determine the energy and the deformed length of the beam and is given by $\|p_2 - p_1\|$. In terms of the global coordinates of the points $p_i = (p_i^{x_g}, p_i^{y_g}, p_i^{z_g})$ with $i = 1, 2$ we can write the gradient as follows

$$\frac{\partial Z}{\partial p_1^\xi} = \frac{p_1^\xi - p_2^\xi}{Z}\,; \quad \frac{\partial Z}{\partial p_2^\xi} = \frac{p_2^\xi - p_1^\xi}{Z} \quad \text{with} \quad \xi \in \{x_g, y_g, z_g\}$$

We also have the following Jacobian matrices in global coordinates

$$\frac{\mathrm{d}\mathbf{e}_z}{\mathrm{d}p_1} = \left(\frac{(p_2-p_1)(p_2-p_1)^\top}{Z^3} - \mathbb{I}_3\frac{1}{Z}\right) \tag{7.1e}$$

$$\frac{\mathrm{d}\mathbf{e}_z}{\mathrm{d}p_2} = \left(\mathbb{I}_3\frac{1}{Z} - \frac{(p_2-p_1)(p_2-p_1)^\top}{Z^3}\right) \tag{7.1f}$$

Here, $\mathbb{I}_3$ is the identity matrix in $\mathbb{R}^{3\times3}$ and $p_i p_j^\top$ denotes the outer product of vectors that results in a rank-one matrix. Thus, the z coordinate of any vector v in the local coordinate system will have the following gradient vectors w.r.t. the global coordinates of p_1 and p_2

$$\frac{\partial v^z}{\partial p_1} = \begin{bmatrix} v^{x_g} & v^{y_g} & v^{z_g} \end{bmatrix} \frac{\mathrm{d}\mathbf{e}_z}{\mathrm{d}p_1} \tag{7.2a}$$

$$\frac{\partial v^z}{\partial p_2} = \begin{bmatrix} v^{x_g} & v^{y_g} & v^{z_g} \end{bmatrix} \frac{\mathrm{d}\mathbf{e}_z}{\mathrm{d}p_2} \tag{7.2b}$$

The other two local coordinates $\xi = x, y$ of any vector v will have the gradient vectors

$$\frac{\partial v^\xi}{\partial p_1} = \begin{bmatrix} v^{x_g} & v^{y_g} & v^{z_g} \end{bmatrix} \frac{\mathrm{d}\mathbf{e}_\xi}{\mathrm{d}\mathbf{e}_z} \frac{\mathrm{d}\mathbf{e}_z}{\mathrm{d}p_1} \tag{7.2c}$$

$$\frac{\partial v^\xi}{\partial p_2} = \begin{bmatrix} v^{x_g} & v^{y_g} & v^{z_g} \end{bmatrix} \frac{\mathrm{d}\mathbf{e}_\xi}{\mathrm{d}\mathbf{e}_z} \frac{\mathrm{d}\mathbf{e}_z}{\mathrm{d}p_2} \tag{7.2d}$$

The orientations of the two end-pieces are given as two quaternions Q_1 and Q_2. These are transformed into tangent unit vectors $\widehat{t}_1$ and $\widehat{t}_2$ that have slopes t_i^ξ in the local direction $\xi = x,$ or y about the local z axis with $i = 1, 2$. Thus

$$t_i^\xi = \frac{\widehat{t}_i^\xi}{\widehat{t}_i^z}$$

and

$$\frac{\partial t_i^\xi}{\partial \widehat{t}_i^\xi} = \frac{1}{\widehat{t}_i^z}\,; \qquad \frac{\partial t_i^\xi}{\partial \widehat{t}_i^z} = -\frac{\widehat{t}_i^\xi}{\widehat{t}_i^{z2}}$$

The respective Jacobian matrices for $i = 1, 2$ in local coordinates are then

$$\frac{\partial t_i}{\partial \widehat{t}_i} = \frac{1}{\widehat{t}_i^z} \begin{bmatrix} 1 & 0 & -\frac{t_i^x}{\widehat{t}_i^z} \\ 0 & 1 & -\frac{t_i^y}{\widehat{t}_i^z} \end{bmatrix}$$

The transformation $Q_i = [Q_i^w, Q_i^x, Q_i^y, Q_i^z]$ to $\widehat{t}_i$ is the action of the quaternion on a unit vector as through the rotation matrix given in (4.1). In the case of the tangent vector we assume this action of the quaternion to be on the global basis unit vector $\mathbf{e}_z = (0, 0, 1)$ and $-\mathbf{e}_z = (0, 0, -1)$ respectively. Hence, we have the

global coordinates

$$\begin{array}{rcl} \widehat{t}_1^{x_g} &=& 2(Q_1^x Q_1^z + Q_1^y Q_1^w) \\ \widehat{t}_1^{y_g} &=& 2(Q_1^y Q_1^z - Q_1^x Q_1^w) \\ \widehat{t}_1^{z_g} &=& Q_1^{z2} + Q_1^{w2} - Q_1^{x2} - Q_1^{y2} \end{array} \quad \text{and} \quad \begin{array}{rcl} \widehat{t}_2^{x_g} &=& -2(Q_2^x Q_2^z + Q_2^y Q_2^w) \\ \widehat{t}_2^{y_g} &=& -2(Q_2^y Q_2^z - Q_2^x Q_2^w) \\ \widehat{t}_2^{z_g} &=& Q_2^{x2} + Q_2^{y2} - Q_2^{z2} - Q_2^{w2} \end{array}$$

giving the respective Jacobian matrices as

$$\frac{\partial \widehat{t}_1^g}{\partial Q_1} = \begin{bmatrix} 2Q_1^y & 2Q_1^z & 2Q_1^w & 2Q_1^x \\ -2Q_1^x & -2Q_1^w & 2Q_1^z & 2Q_1^y \\ 2Q_1^w & -2Q_1^x & -2Q_1^y & 2Q_1^z \end{bmatrix}$$

and

$$\frac{\partial \widehat{t}_2^g}{\partial Q_2} = \begin{bmatrix} -2Q_2^y & -2Q_2^z & -2Q_2^w & -2Q_2^x \\ 2Q_2^x & 2Q_2^w & -2Q_2^z & -2Q_2^y \\ -2Q_2^w & 2Q_2^x & 2Q_2^y & -2Q_2^z \end{bmatrix}$$

Due to transformation of coordinates from global to local we know

$$\frac{\mathrm{d}\widehat{t}_i}{\mathrm{d}\widehat{t}_i^g} = \begin{bmatrix} \mathbf{e}_x^\top \\ \mathbf{e}_y^\top \\ \mathbf{e}_z^\top \end{bmatrix} \qquad \text{since} \qquad \widehat{t}_i^\top = [\ \widehat{t}_i^{x_g} \quad \widehat{t}_i^{y_g} \quad \widehat{t}_i^{z_g}\ ][\ \mathbf{e}_x \quad \mathbf{e}_y \quad \mathbf{e}_z\ ]$$

where $\mathbf{e}_\xi$ are written as column vectors in their global coordinates. Multiplying the above matrices we get the Jacobian matrices for the mapping from the Quaternions to the slopes.

$$\frac{\partial t_1}{\partial Q_1} = \frac{2}{\widehat{t}_1^z} \begin{bmatrix} 1 & 0 & -\frac{\widehat{t}_1^x}{\widehat{t}_1^z} \\ 0 & 1 & -\frac{\widehat{t}_1^y}{\widehat{t}_1^z} \end{bmatrix} \begin{bmatrix} \mathbf{e}_x^\top \\ \mathbf{e}_y^\top \\ \mathbf{e}_z^\top \end{bmatrix} \begin{bmatrix} Q_1^y & Q_1^z & Q_1^w & Q_1^x \\ -Q_1^x & -Q_1^w & Q_1^z & Q_1^y \\ Q_1^w & -Q_1^x & -Q_1^y & Q_1^z \end{bmatrix}$$

$$\frac{\partial t_2}{\partial Q_2} = -\frac{2}{\widehat{t}_2^z} \begin{bmatrix} 1 & 0 & -\frac{\widehat{t}_2^x}{\widehat{t}_2^z} \\ 0 & 1 & -\frac{\widehat{t}_2^y}{\widehat{t}_2^z} \end{bmatrix} \begin{bmatrix} \mathbf{e}_x^\top \\ \mathbf{e}_y^\top \\ \mathbf{e}_z^\top \end{bmatrix} \begin{bmatrix} Q_2^y & Q_2^z & Q_2^w & Q_2^x \\ -Q_2^x & -Q_2^w & Q_2^z & Q_2^y \\ Q_2^w & -Q_2^x & -Q_2^y & Q_2^z \end{bmatrix}$$

The upper row in the resulting Jacobians correspond to $\xi = x$ and is denoted $\frac{\partial t_i^x}{\partial Q_i}$ whereas the lower row corresponds to $\xi = y$ and is denoted $\frac{\partial t_i^x}{\partial Q_i}$ with $i = 1, 2$. In the expression for the energy of a single beam these slopes occur in the following form

$$(t_1^\xi + t_2^\xi)^2 \quad \text{or} \quad (t_2^\xi - t_1^\xi)^2$$

and the corresponding gradients can be written as the row vectors

$$\frac{\partial(t_1^\xi + t_2^\xi)^2}{\partial\{Q_1, Q_2\}} = 2(t_1^\xi + t_2^\xi)\left[\begin{array}{cc} \dfrac{\partial t_1^\xi}{\partial Q_1}, & \dfrac{\partial t_2^\xi}{\partial Q_2} \end{array}\right]$$

$$\frac{\partial(t_2^\xi - t_1^\xi)^2}{\partial\{Q_1, Q_2\}} = 2(t_2^\xi - t_1^\xi)\left[\begin{array}{cc} -\dfrac{\partial t_1^\xi}{\partial Q_1}, & \dfrac{\partial t_2^\xi}{\partial Q_2} \end{array}\right]$$

Also, using each derivative in (7.1) and (7.2) we have the following gradient vectors

$$\frac{\mathrm{d}t_i^\xi}{\mathrm{d}p_j} = \frac{\partial t_i^\xi}{\partial \widehat{t}_i^\xi}\frac{\mathrm{d}\widehat{t}_i^\xi}{\mathrm{d}p_j} + \frac{\partial t_i^\xi}{\partial \widehat{t}_i^z}\frac{\mathrm{d}\widehat{t}_i^z}{\mathrm{d}p_j} \qquad \text{for} \quad \xi \in \{x, y\}\ i, j \in \{1, 2\}$$

and thus, the vectors

$$\frac{\partial(t_1^\xi + t_2^\xi)^2}{\partial\{p_1, p_2\}} = 2(t_1^\xi + t_2^\xi)\left[\begin{array}{cc} \dfrac{\mathrm{d}t_1^\xi}{\mathrm{d}p_1} + \dfrac{\mathrm{d}t_2^\xi}{\mathrm{d}p_1}, & \dfrac{\mathrm{d}t_1^\xi}{\mathrm{d}p_2} + \dfrac{\mathrm{d}t_2^\xi}{\mathrm{d}p_2} \end{array}\right]$$

$$\frac{\partial(t_2^\xi - t_1^\xi)^2}{\partial\{p_1, p_2\}} = 2(t_2^\xi - t_1^\xi)\left[\begin{array}{cc} -\dfrac{\mathrm{d}t_1^\xi}{\mathrm{d}p_1} + \dfrac{\mathrm{d}t_2^\xi}{\mathrm{d}p_1}, & -\dfrac{\mathrm{d}t_1^\xi}{\mathrm{d}p_2} + \dfrac{\mathrm{d}t_2^\xi}{\mathrm{d}p_2} \end{array}\right]$$

7.2 From local state to energy

From (4.16) we have the unique axial force P as the solution of (4.17) Thus, the axial force P is an implicit function of the global state variables $\mathcal{P}$ and the derivative of the energy with respect to the state variables can be written using the implicit function theorem as follows

$$\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}\mathcal{P}}(P(\mathcal{P}), \mathcal{P}) = \frac{\partial\mathcal{E}}{\partial\mathcal{P}} + \frac{\partial\mathcal{E}}{\partial P}\frac{\mathrm{d}P}{\mathrm{d}\mathcal{P}}$$

The local state is defined completely in terms of the local variables Z and t_i^ξ along with the intermediate unknown P. Thus in order to find the derivatives of the energy with respect to the local variables we only use the expressions for the energy as computed in (4.14) and (4.15). Since $\mathcal{P}$ are the global state variables let us call $\mathcal{P}_{\text{loc}}$ the local state variables (Z, t_i^ξ) with $i = 1, 2$ and $\xi = x, y$. Components of the Jacobian $\dfrac{\mathrm{d}\mathcal{P}_{\text{loc}}}{\mathrm{d}\mathcal{P}}$ have already been computed in the previous section. So the above derivative can be written as

$$\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}\mathcal{P}}(P(\mathcal{P}), \mathcal{P}) = \left[\frac{\partial\mathcal{E}}{\partial\mathcal{P}_{\text{loc}}} + \frac{\partial\mathcal{E}}{\partial P}\frac{\mathrm{d}P}{\mathrm{d}\mathcal{P}_{\text{loc}}}\right]\frac{\mathrm{d}\mathcal{P}_{\text{loc}}}{\mathrm{d}\mathcal{P}}$$

Since the series involved are reformulated in terms of W we may eliminate the variable P and rewrite

$$\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}\mathcal{P}}(W(\mathcal{P}),\mathcal{P}) = \left[\frac{\partial\mathcal{E}}{\partial\mathcal{P}_{\mathrm{loc}}} + \frac{\partial\mathcal{E}}{\partial W}\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}\right]\frac{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}{\mathrm{d}\mathcal{P}} \tag{7.3}$$

Let us now write the expression for $\dfrac{\partial\mathcal{E}}{\partial\mathcal{P}_{\mathrm{loc}}}$ from (4.14) and (4.15) and $\dfrac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}$ from (4.17) using the implicit function theorem.

For the second term of the energy derivative (7.3) we differentiate (4.17) with respect to $\mathcal{P}_{\mathrm{loc}}$ component-wise at the solution point $W_*(\mathcal{P}_{\mathrm{loc}})$, considering it an implicit function

$$\begin{aligned}-2\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}\frac{\partial\widetilde{L}}{\partial W} - W\frac{\partial}{\partial\mathcal{P}_{\mathrm{loc}}}\left(\frac{\partial\widetilde{L}}{\partial W}\right) - W\frac{\partial^2\widetilde{L}}{\partial W^2}\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}} - \frac{\partial\widetilde{L}}{\partial\mathcal{P}_{\mathrm{loc}}}\\ - \frac{4IL}{AZ^2}\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}} - \frac{4ILW}{A}\frac{\mathrm{d}}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}\left(\frac{1}{Z^2}\right) = 0\end{aligned}$$

with $\dfrac{\partial\widetilde{L}}{\partial W}$ as in (4.18) and

$$\frac{\partial^2\widetilde{L}}{\partial W^2} = [(t_1^x+t_2^x)^2+(t_1^y+t_2^y)^2]\frac{Z}{2}\frac{\partial^2\widetilde{L}_{\mathrm{odd}}}{\partial W^2} + [(t_2^x-t_1^x)^2+(t_2^y-t_1^y)^2]\frac{Z}{2}\frac{\partial^2\widetilde{L}_{\mathrm{even}}}{\partial W^2}$$

Gathering the terms containing $\dfrac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}$ yields:

$$\begin{aligned}\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}\left[2\frac{\partial\widetilde{L}}{\partial W} + W\frac{\partial^2\widetilde{L}}{\partial W^2} + \frac{4IL}{AZ^2}\right] = \Bigg[- W\frac{\partial}{\partial\mathcal{P}_{\mathrm{loc}}}\left(\frac{\partial\widetilde{L}}{\partial W}\right) - \frac{\partial\widetilde{L}}{\partial\mathcal{P}_{\mathrm{loc}}}\\ - \frac{4ILW}{A}\frac{\mathrm{d}}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}\left(\frac{1}{Z^2}\right)\Bigg]\end{aligned}$$

This results in the following gradient at W_*

$$\begin{aligned}\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}} = \Bigg[- W\frac{\partial}{\partial\mathcal{P}_{\mathrm{loc}}}\left(\frac{\partial\widetilde{L}}{\partial W}\right) - \frac{\partial\widetilde{L}}{\partial\mathcal{P}_{\mathrm{loc}}}\\ - \frac{4ILW}{A}\frac{\mathrm{d}}{\mathrm{d}\mathcal{P}_{\mathrm{loc}}}\left(\frac{1}{Z^2}\right)\Bigg]\left[2\frac{\partial\widetilde{L}}{\partial W} + W\frac{\partial^2\widetilde{L}}{\partial W^2} + \frac{4IL}{AZ^2}\right]^{-1}\end{aligned} \tag{7.4}$$

where the right hand side is evaluated component-wise. In chapter 2 it was described how this implicit derivative can be calculated with the *nested differentiation* feature of `ADTAGEO` automatically.

For the derivatives of the form $\frac{\partial}{\partial \mathcal{P}_{\text{loc}}}$ we need their components $\frac{\partial}{\partial (t_2^\xi \pm t_1^\xi)^2}$ and $\frac{\partial}{\partial Z}$ computed at W_* as follows:

$$\left.\frac{\partial \widetilde{L}}{\partial (t_1^\xi + t_2^\xi)^2}\right|_{W_*} = \frac{Z}{2}\left.\widetilde{L}_{\text{odd}}\right|_{W_*}$$

$$\left.\frac{\partial \widetilde{L}}{\partial (t_2^\xi - t_1^\xi)^2}\right|_{W_*} = \frac{Z}{2}\left.\widetilde{L}_{\text{even}}\right|_{W_*}$$

$$\left.\frac{\partial}{\partial (t_1^\xi + t_2^\xi)^2}\left(\frac{\partial \widetilde{L}}{\partial W}\right)\right|_{W_*} = \frac{Z}{2}\left.\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W}\right|_{W_*}$$

$$\left.\frac{\mathrm{d}}{\mathrm{d}(t_2^\xi - t_1^\xi)^2}\left(\frac{\partial \widetilde{L}}{\partial W}\right)\right|_{W_*} = \frac{Z}{2}\left.\frac{\partial \widetilde{L}_{\text{even}}}{\partial W}\right|_{W_*}$$

$$\begin{aligned}\left.\frac{\partial \widetilde{L}}{\partial Z}\right|_{W_*} = 1 &+ [(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\left.\frac{1}{2}\widetilde{L}_{\text{odd}}\right|_{W_*} \\ &+ [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\left.\frac{1}{2}\widetilde{L}_{\text{even}}\right|_{W_*}\end{aligned}$$

$$\begin{aligned}\left.\frac{\partial}{\partial Z}\left(\frac{\partial \widetilde{L}}{\partial W}\right)\right|_{W_*} = &[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\frac{1}{2}\left.\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W}\right|_{W_*} \\ &+ [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\frac{1}{2}\left.\frac{\partial \widetilde{L}_{\text{even}}}{\partial W}\right|_{W_*}\end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{d}(t_2^\xi \pm t_1^\xi)^2}\left(\frac{1}{Z^2}\right) = 0$$

$$\frac{\mathrm{d}}{\mathrm{d}Z}\left(\frac{1}{Z^2}\right) = -\frac{2}{Z^3}$$

Rewriting (4.15) in terms of W we get

$$\mathcal{E}_l = -\frac{8EI^2W^2L}{AZ^4} - \frac{4EIW}{Z^2}(\widetilde{L} - L) \tag{7.5}$$

along with the derivative

$$\frac{\partial \mathcal{E}_l}{\partial W} = -\frac{16EI^2WL}{AZ^4} - \frac{4EI}{Z^2}(\widetilde{L} - L) - \frac{4EIW}{Z^2}\frac{\partial \widetilde{L}}{\partial W}$$

$$= \frac{4EI}{Z^2}\left[-\frac{4IWL}{AZ^2} - \widetilde{L} + L - W\frac{\partial \widetilde{L}}{\partial W}\right]$$

resulting in

$$\frac{\partial \mathcal{E}}{\partial W} = \frac{\partial \mathcal{E}_b}{\partial W} + \frac{4EI}{Z^2}\left[-\frac{4IWL}{AZ^2} - \widetilde{L} + L - W\frac{\partial \widetilde{L}}{\partial W}\right] \tag{7.6}$$

The partial derivatives w.r.t. W at W_* are

$$\begin{aligned}\left.\frac{\partial \widetilde{L}}{\partial W}\right|_{W_*} =& [(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\frac{Z}{2}\left.\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W}\right|_{W_*} \\ &+ [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\frac{Z}{2}\left.\frac{\partial \widetilde{L}_{\text{even}}}{\partial W}\right|_{W_*}\end{aligned}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{E}_b}{\partial W}\right|_{W_*} =& [(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\ \frac{EI}{Z}\left.\frac{\partial \mathcal{E}_{b\text{odd}}}{\partial W}\right|_{W_*} \\ &+ [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\ \frac{EI}{Z}\left.\frac{\partial \mathcal{E}_{b\text{even}}}{\partial W}\right|_{W_*}\end{aligned}$$

Evaluating $\dfrac{\partial \mathcal{E}}{\partial W}$ from (7.6) by substituting the above two expressions we see

$$\left.\frac{\partial \mathcal{E}}{\partial W}\right|_{W_*} = \left.\frac{\partial \mathcal{E}_b}{\partial W}\right|_{W_*} + \frac{4EI}{Z^2}\widetilde{\mathcal{F}}(W_*)$$

If we solve $\widetilde{\mathcal{F}}(W) = 0$ accurately then this part of the derivative vanishes. Hence the computation of the second term of the state derivative of the energy is somewhat simplified. An inexact solution of $\widetilde{\mathcal{F}}(W) = 0$, however, will not have this property. If one uses `ADTAGEO` one may solve the simpler $\mathcal{F}(P) = 0$ or $\widehat{\mathcal{F}}(P) = 0$ and rely on the computation of *nested derivatives*. Therefore one must solve $\widetilde{F}(W) = 0$ accurately if one wishes to simplify the state derivative computation without the use of `ADTAGEO`.

Having found the implicit derivative in (7.4) and the partial derivative w.r.t. W in (7.6) we have the second term of the sum in (7.3). Now one only needs to differentiate the explicit dependence of the energy on the local state $\mathcal{P}_{\text{loc}}$ at the optimal W_* for the first term of the sum. From (4.14) first we write

$$\left.\frac{\partial \mathcal{E}_b}{\partial (t_1^\xi + t_2^\xi)^2}\right|_{W_*} = \frac{EI}{Z}\ \mathcal{E}_{b\text{odd}}|_{W_*}$$

$$\left.\frac{\partial \mathcal{E}_b}{\partial (t_2^\xi - t_1^\xi)^2}\right|_{W_*} = \frac{EI}{Z}\, \mathcal{E}_{b\text{even}}|_{W_*}$$

$$\left.\frac{\partial E_b}{\partial Z}\right|_{W_*} = -\frac{EI}{Z^2}\Big[[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\; \mathcal{E}_{b\text{odd}}|_{W_*} + [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\; \mathcal{E}_{b\text{even}}|_{W_*}\Big]$$

Differentiating (7.5) w.r.t. $\mathcal{P}_{\text{loc}}$ at W_* we get

$$\frac{\partial \mathcal{E}_l}{\partial \mathcal{P}_{\text{loc}}} = -\frac{8EI^2 W_*^2 L}{A}\frac{\partial}{\partial \mathcal{P}_{\text{loc}}}\left(\frac{1}{Z^4}\right) - 4EIW_*(\widetilde{L}|_{W_*} - L)\frac{\partial}{\partial \mathcal{P}_{\text{loc}}}\left(\frac{1}{Z^2}\right) - \frac{4EIW_*}{Z^2}\left.\frac{\partial \widetilde{L}}{\partial \mathcal{P}_{\text{loc}}}\right|_{W_*}$$

where

$$\frac{\partial}{\partial (t_2^\xi \pm t_1^\xi)}\left(\frac{1}{Z^j}\right) = 0 \quad \xi = x, y \quad j = 2, 4$$

$$\frac{\partial}{\partial Z}\left(\frac{1}{Z^j}\right) = -j\frac{1}{Z^{j+1}} \quad j = 2, 4$$

In the irregular case of $W_* = \pi^2$ and β_{even} chosen as in (4.25) we have $\frac{\mathrm{d}W}{\mathrm{d}\mathcal{P}_{\text{loc}}} = 0$, but β_{even} depends on the local state $\mathcal{P}_{\text{loc}}$ and we must evaluate $\frac{\mathrm{d}\beta_{\text{even}}}{\mathrm{d}\mathcal{P}_{\text{loc}}}$ in order to compute the full derivatives of the energy w.r.t $\mathcal{P}_{\text{loc}}$. Differentiating (4.25) we have

$$\frac{\mathrm{d}\beta_{\text{even}}}{\mathrm{d}Z} = 16\frac{\left[12\frac{IL\pi^2}{AZ^4} - \frac{L}{Z^2}\right]}{\left[\sum_{j=0}^{\infty}(-1)^j(j+1)C^{(2)}\pi^{2j}\right]}$$

$$\frac{\mathrm{d}\beta_{\text{even}}}{\mathrm{d}(t_1^\xi + t_2^\xi)^2} = -8\frac{\left[\pi^2\left.\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W}\right|_{\pi^2} + \left.\widetilde{L}_{\text{odd}}\right|_{\pi^2}\right]}{\left[\sum_{j=0}^{\infty}(-1)^j(j+1)C_j^{(2)}\pi^{2j}\right]}$$

Substituting in the expressions for $\mathcal{E}_b$ and $\widetilde{L}$ as in (4.24)

$$\frac{\mathrm{d}\mathcal{E}_b}{\mathrm{d}Z} = EI\Big[[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\left(-\frac{1}{Z^2}\mathcal{E}_{b\text{odd}}\Big|_{\pi^2}\right) + \left(\frac{1}{Z}\frac{\mathrm{d}\beta_{\text{even}}}{\mathrm{d}Z} - \frac{1}{Z^2}\beta_{\text{even}}\right)\sum_{j=0}^{\infty}(-1)^j C_j^{(3)}\pi^{2j}\Big]$$

$$\frac{\mathrm{d}\mathcal{E}_b}{\mathrm{d}(t_1^\xi + t_2^\xi)^2} = \frac{EI}{Z}\left[\mathcal{E}_{b\mathrm{odd}} + \frac{\mathrm{d}\beta_{\mathrm{even}}}{\mathrm{d}(t_1^\xi + t_2^\xi)^2}\sum_{j=0}^{\infty}(-1)^j C_j^{(3)}\pi^{2j}\right]$$

$$\frac{\mathrm{d}\widetilde{L}}{\mathrm{d}Z} = 1 + \left[[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\left(\frac{1}{2}\,\widetilde{L}_{\mathrm{odd}}\Big|_{\pi^2}\right) + \left(\frac{Z}{2}\frac{\mathrm{d}\beta_{\mathrm{even}}}{\mathrm{d}Z} + \frac{1}{2}\beta_{\mathrm{even}}\right)\sum_{j=0}^{\infty}(-1)^j C_j^{(2)}\pi^{2j}\right]$$

$$\frac{\mathrm{d}\widetilde{L}}{\mathrm{d}(t_1^\xi + t_2^\xi)^2} = \frac{Z}{2}\left[\widetilde{L}_{\mathrm{odd}} + \frac{\mathrm{d}\beta_{\mathrm{even}}}{\mathrm{d}(t_1^\xi + t_2^\xi)^2}\sum_{j=0}^{\infty}(-1)^j C_j^{(2)}\pi^{2j}\right]$$

$$\frac{\mathrm{d}\mathcal{E}_b}{\mathrm{d}(t_2^\xi - t_1^\xi)^2} = 0 \quad \text{and} \quad \frac{\mathrm{d}\widetilde{L}}{\mathrm{d}(t_2^\xi - t_1^\xi)^2} = 0 \quad \text{as} \quad \frac{\mathrm{d}\beta_{\mathrm{even}}}{\mathrm{d}(t_2^\xi - t_1^\xi)^2} = 0$$

7.3 Internal balance of axial force

In order to compute the energy and the derivatives above we require the value of the intermediate unknown P or W, which may be computed either by solving the nonlinear equation (4.20) using the series equation (4.26) for small bending, or by solving the envelope equation (4.17) that is the same as (4.23) as described in chapter 4 via the safeguarded Newton iteration of section 4.6. The required derivative is written as follows

$$\frac{\mathrm{d}\mathcal{F}}{\mathrm{d}P} = \frac{\partial\mathcal{F}}{\partial P} = -\frac{L}{EA} - \frac{Z^2}{8EI}\left\{[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\frac{\partial\widetilde{L}_{\mathrm{odd}}}{\partial W} + [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\frac{\partial\widetilde{L}_{\mathrm{even}}}{\partial W}\right\}$$

$$\begin{aligned}\frac{\mathrm{d}\widehat{\mathcal{F}}}{\mathrm{d}P} = \frac{\partial\widehat{\mathcal{F}}}{\partial P} = &-\frac{L}{EA}\sum_{j=0}^{\infty}\widehat{C}_j^{(3)}W^j + \\ &\left(L - Z - \frac{PL}{EA}\right)\sum_{j=0}^{\infty}(j+1)\widehat{C}_{j+1}^{(3)}W^j\frac{\partial W}{\partial P} \\ &-\frac{Z}{16}\left[\frac{1}{4}[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\sum_{j=0}^{\infty}(j+1)\widehat{C}_{j+1}^{(1)}W^j \right. \\ &\left. + [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\sum_{j=0}^{\infty}(j+1)\widehat{C}_{j+1}^{(2)}W^j\right]\frac{\partial W}{\partial P}\end{aligned}$$

with the series coefficients as defined in chapter 4.

$$\begin{aligned}\frac{\mathrm{d}\widetilde{\mathcal{F}}}{\mathrm{d}W} =& Z\Bigg[[(t_1^x + t_2^x)^2 + (t_1^y + t_2^y)^2]\left\{-\frac{\partial \widetilde{L}_{\text{odd}}}{\partial W} - \frac{W}{2}\frac{\partial^2 \widetilde{L}_{\text{odd}}}{\partial W^2}\right\} \\ &+ [(t_2^x - t_1^x)^2 + (t_2^y - t_1^y)^2]\left\{-\frac{\partial \widetilde{L}_{\text{even}}}{\partial W} - \frac{W}{2}\frac{\partial^2 \widetilde{L}_{\text{even}}}{\partial W^2}\right\}\Bigg] \\ &- \frac{4IL}{AZ^2}\end{aligned}$$

We assume, as mentioned before, that each of $\mathcal{F}(P)$, $\widehat{\mathcal{F}}(P)$ and $\widetilde{\mathcal{F}}(W)$ is evaluated for constant Z and tangents. One may note that and $\mathcal{F}$ and $\widetilde{\mathcal{F}}$ are monotonically decreasing functions and have a negative pole at $W = \pi^2$ or $P = \frac{4EI\pi^2}{Z^2}$.

7.4 Series expansions and their derivatives

For each series expansion as seen in chapter 4 a modified Horner scheme for computation is as follows. We rewrite

$$\sum_{j=0}^{\infty} c_j W^j = \sum_{j=0}^{\infty} a_j W^j \prod_{k=0}^{j} b_k \quad \text{s.t.} \;\; c_j = a_j \prod_{k=0}^{j} b_k$$

In this form a truncated sum can be evaluated, as also mentioned in chapter 4, in the following form

$$b_0(a_0 + b_1 W(a_1 + b_2 W(a_2 + \ldots (a_{j-1} + b_j W(a_j + \ldots)\ldots))))$$

along with its derivative w.r.t. W

$$b_0 b_1(a_1 + b_2 W(2a_2 + \ldots ((j-1)a_{j-1} + b_j W(j a_j + \ldots)\ldots))))$$

and the second derivative w.r.t. W

$$b_0 b_1 b_2(2a_2 + b_3 W(2 \cdot 3a_3 + \ldots ((j-2)(j-1)a_{j-1} + b_{j-1}((j-1)j a_j + \ldots)\ldots)))$$

The truncation index $j_{\max}$ may be found for each series expansion with the help of Stirling's formula as discussed in chapter 4.

For each of the series involved in the numerator and denominator of $\widetilde{L}_{\text{odd}}$, $\widetilde{L}_{\text{even}}$ and $\mathcal{E}_{\text{odd}}$, $\mathcal{E}_{\text{even}}$ the size of the j^{th} term (T_j) can be bounded from above by

the following

$$|T_j| \leqslant \left| \frac{2^{2j} W^j}{(2j)!} \right| \approx \frac{1}{\sqrt{4\pi j}} \left| \frac{e^2 W}{j^2} \right|^j$$

The maximal value $j_{\max}$ to bound the exponential part by any given $\varepsilon > 0$ can be written as

$$j_{\max} = -\frac{1}{2} \ln \varepsilon \left[\mathcal{W} \left(\frac{-\ln \varepsilon}{2e\sqrt{|W|}} \right) \right]^{-1}$$

where $\mathcal{W}$ is the Lambert $\mathcal{W}$ function [CORLESS et al., 1996].

Chapter 8

Numerics and observations

In this Chapter the numerical results obtained from an implementation of the beam and frame model in Standard C++ are presented. The code was developed independently and without the use of any other modelling software. The only software dependency is the Standard C++ library from the GNU Compiler Collection that is licensed under the GNU General Public License version 2, with the so-called "Runtime Exception" as follows:

> As a special exception, you may use this file as part of a free software library without restriction. Specifically, if other files instantiate templates or use macros or inline functions from this file, or you compile this file and link it with other files to produce an executable, this file does not by itself cause the resulting executable to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the executable file might be covered by the GNU General Public License.

The plots and figures presented in this Chapter have been prepared using either Matlab® or `gnuplot` from the output of the C++ code. The C++ code is single threaded and sequential although potential parallelism exists, and all the CPU timings quoted were on an Intel® Core 2^{TM} 64-bit 2.83 GHz processor with 6MB Cache under Linux (kernel v2.6.27 x86_64). The C++ code was developed only as a numerical validation tool applied to small examples of academic nature for the theory presented in the previous Chapters. It is not meant for distribution, external publication or commercial use.

8.1 Single beam

As described in chapter 4, the energy of the beam has a complex dependence on the variables W and Z. Table 8.1 lists the Taylor series coefficients for $\widetilde{L}_{\text{odd}\,|\,\text{even}}$ upto order 15. These are all positive and diminish rapidly with j and corroborate Proposition 4.1. In Figure 8.1 the variation of the total elastic energy is

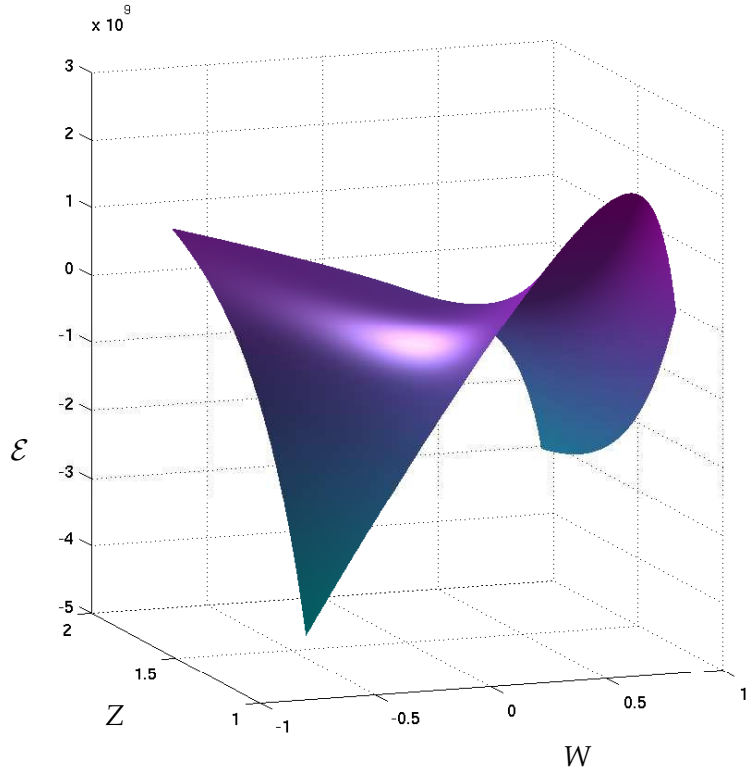


Figure 8.1: The dependence of the energy of a single beam with specified non-trivial tangent directions and length L with changing W and Z.

plotted against the variation of these two variables. We recognise the stable state for the given natural length L and tangents t_i^{ξ} is a saddle point, with a maximum in W and a minimum in Z. The energy of a single beam also depends on the given tangent directions in a complicated manner. In Figure 8.2 this dependence for given L at W_* is depicted as Z changes from large compression to elongation. One can notice a non-smooth high energy maximal ridge along the diagonal $t_1 = t_2$ whenever Euler-buckling happens. As Z increases, thereby reducing the amount of compression the beam undergoes, this ridge becomes smaller and smaller in size, becoming smooth for compression which is small enough to not cause buckling, and the two minima on its either side come progressively closer. Since the variation of tangents is only in one plane in Figure 8.2 it verifies the observation that it is impossible to go smoothly from one minimum to the other of the buckled state within this plane; to do so the beam would have to deform in the third dimension. When no buckling happens it is possible to go smoothly from one minimum to the other. Further-

more the minima both lie on the negative diagonal $t_1 = -t_2$ and favour the "U" shape over the "S" shape. For elongation the ridge vanishes completely and the two minima become one at the origin. This verifies the observation that for elongation the straighter the beam the less energy it has.

j	Taylor coefficient for $\widetilde{L}_{\text{odd}}$	j	Taylor coefficient for $\widetilde{L}_{\text{even}}$
0	$\frac{1}{20}$	0	$\frac{1}{12}$
1	$\frac{1}{350}$	1	$\frac{1}{90}$
2	$\frac{1}{5250}$	2	$\frac{1}{630}$
3	$\frac{37}{3031875}$	3	$\frac{1}{4725}$
4	$\frac{59}{78828750}$	4	$\frac{1}{37422}$
5	$\frac{2753}{62077640625}$	5	$\frac{691}{212837625}$
6	$\frac{827}{323057109375}$	6	$\frac{1}{2606175}$
7	$\frac{16772918}{115794974998828125}$	7	$\frac{7234}{162820783125}$
8	$\frac{28033727}{3473849249964843750}$	8	$\frac{43867}{8662065662250}$
9	$\frac{14529522883}{3271844916079388085937 5}$	9	$\frac{174611}{306265893058125}$
10	$\frac{163546417}{6760010157188818359375}$	10	$\frac{77683}{1222532449149375}$
11	$\frac{94994770034}{726727091936864084472656 25}$	11	$\frac{472728182}{67306523987918840625}$
12	$\frac{170551314127}{243174065378873751342773 4375}$	12	$\frac{657931}{853421690463890625}$
13	$\frac{4050931213731634}{108289058924193164579207153320 3125}$	13	$\frac{6785560294}{80664808595725181953125}$
14	$\frac{528166904630274578}{266066217776742605371111975708007 8125}$	14	$\frac{3446336510402}{37739192031127217827133 4375}$
15	$\frac{3061635482015854866172}{291941157455530823743452615345611572265625}$	15	$\frac{30837284164868}{31245110285511170603633203125}$

Table 8.1: Taylor coefficients for $\widetilde{L}_{\text{odd}\,|\,\text{even}}$ upto order 15

It is interesting to observe the numerical computation in Figure 8.2, in the third column, that the optimal W_* for a beam is smooth w.r.t. the state variables

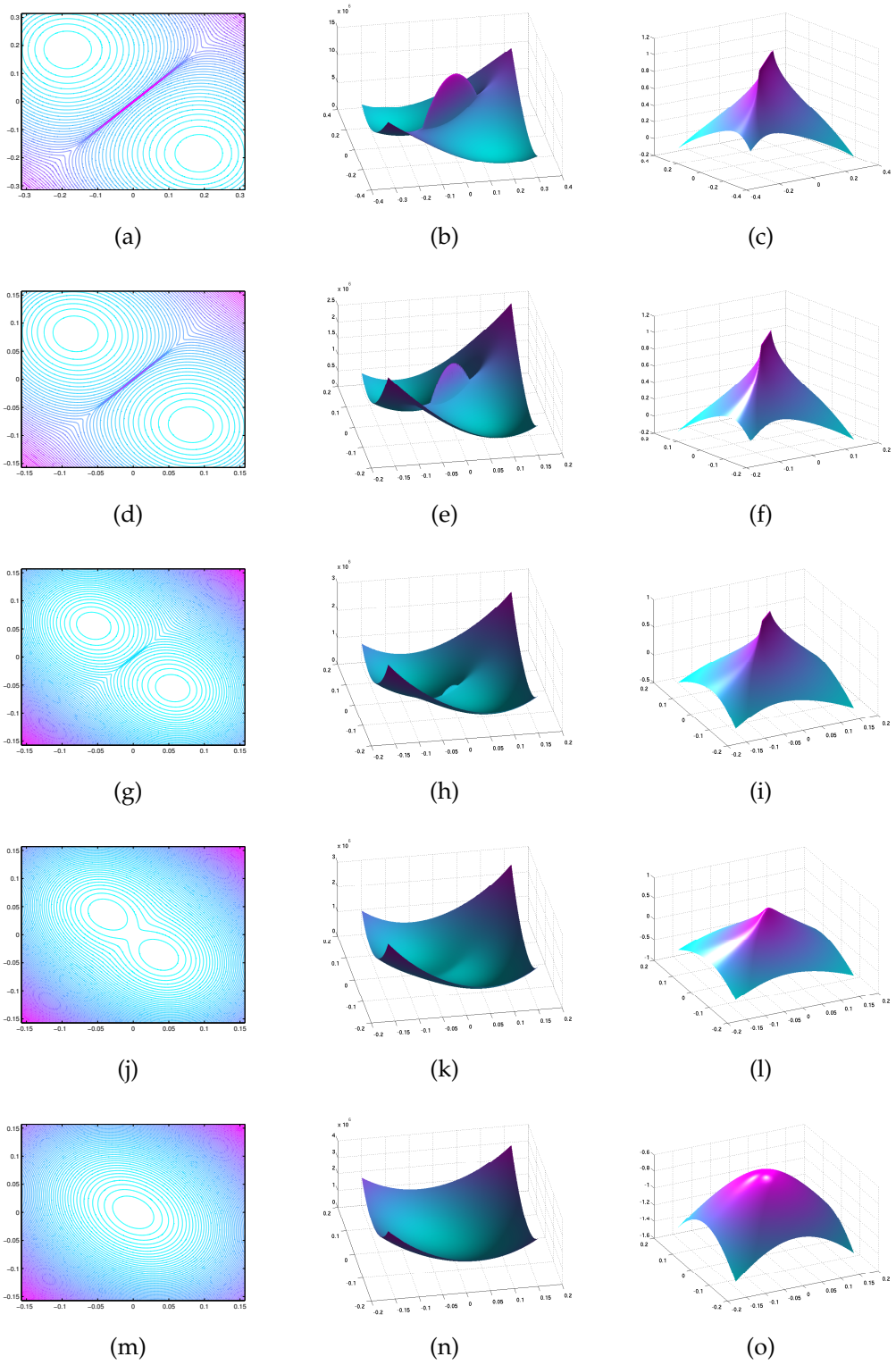


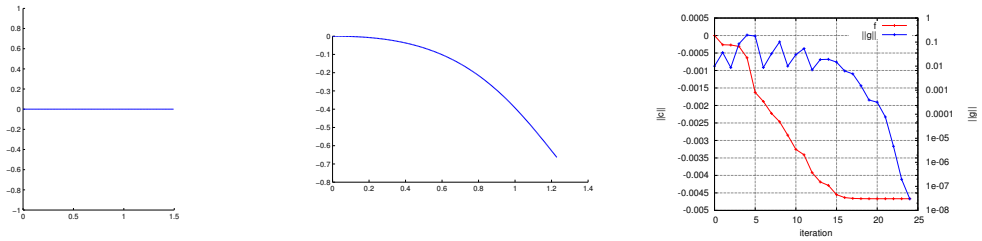
Figure 8.2: Variation of energy and W_* with tangents t_1 and t_2 in one plane for given $L = 1.5$ as Z changes; the first two columns show the total energy the third column shows W_* in multiples of π^2; (a), (b), (c) $Z = 1.48$; (d), (e), (f) $Z = 1.496$; (g), (h), (i) $Z = 1.498$; (j), (k), (l) $Z = 1.499$; (m), (n), (o) $Z = 1.501$.

except when the irregular case (see page 32 and page 39) is encountered. In the irregular case W_* appears to be Lipschitz. This corroborates the result of Proposition 4.2.

8.2 Stable states of simple frames

The the following results were obtained using the software implementation `LRAMBO` of the total quasi Newton method GRIEWANK et al. [2004–2010, 2007b]. As discussed in chapter 3 this maintains factorisations of Hessian and Jacobian approximation matrices, requiring only gradient vectors or directional derivatives exactly, along with a Shift and Zoom linesearch [KREITERLING, 2007] in order to do the unconstrained minimisation as in section 5.4.

Example 8.1 (Cantilever beam)**.** We have a single beam of crossection 0.754 cm^2 with one end fixed both in its position $(0, 0)$ and its orientation and the other end $(1.5, 0)$ free to move and rotate. A large orthogonal load $(0, -10^3)$ N is applied at the free end (Figure 8.3). The problem has 4 variables and the BFGS method required 24 iterations starting with a multiple of the identity as the initial Hessian approximation and 272 energy evaluations and takes 0.024s of CPU time (0.444s when using `ADTAGEO`).

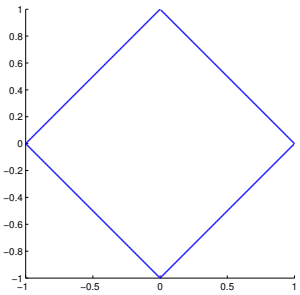


(a) undeformed state (b) deformed state (c) iteration history

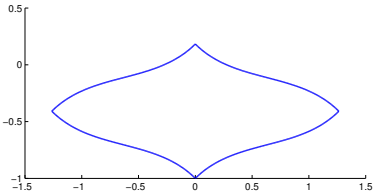
Figure 8.3: Cantilever beam

Example 8.2. Consider a rhombus shaped frame made of beams with equal crossection (0.754 cm^2) with the bottommost joint completely fixed in its position and orientation. Each joint is rigid, i.e. the respective angles do not change. All the three other points are free to move in the plane of the rhombus as well as rotate in this plane. The problem has 12 variables. Loads are applied at the topmost point. The solution method is the same as before with initial Hessian being a multiple of the identity. Figure 8.4 shows the undeformed and deformed

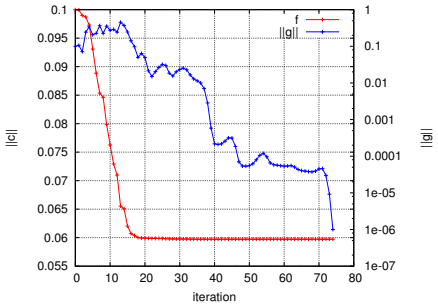
states as well as the iteration history. Convergence to stable state in Figure 8.4b requires 74 iterations (Figure 8.4c) with 3448 individual beam energy evaluations and CPU time of 0.152s (4.584s when using `ADTAGE0`). Convergence to stable state in Figure 8.4d requires 116 iterations (Figure 8.4e) with 5576 individual beam energy evaluations and CPU time of 0.216s (8.901s when using `ADTAGE0`). This planar example highlights the effect of rigid joints between the beams, without which the rhombus shaped frame would collapse even under its own weight instead of withstanding the applied forces at the top.



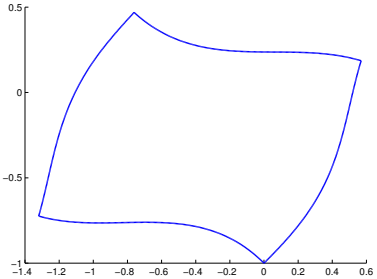
(a) undeformed



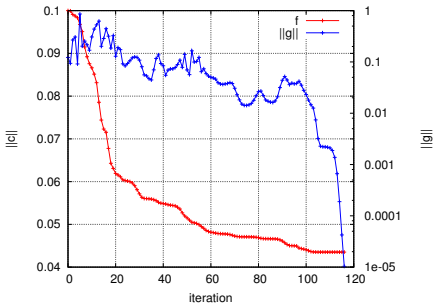
(b) load $(0, -10^4)$ N



(c) iteration history



(d) load $(-7 \times 10^3, -10^4)$ N

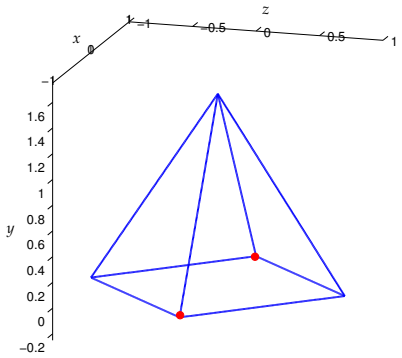


(e) iteration history

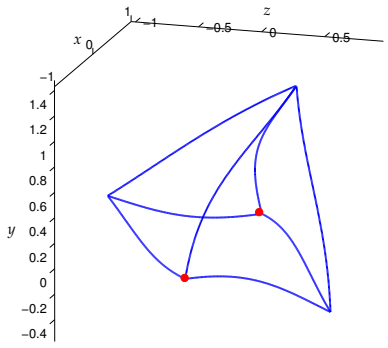
Figure 8.4: Rhombus shaped frame

Example 8.3. Consider a pyramid with square base and rigid joints that has two of its base points fixed (8 beams connecting 2 fixed and 3 free Frame-

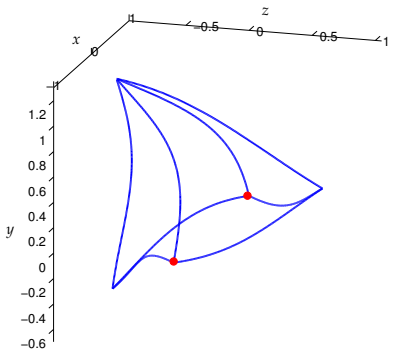
points). The problem has 21 variables and is not planar but must be computed in 3 dimensions. Figure 8.5 shows two symmetric load cases with the load applied at the apex of the pyramid. The fixed base points in the figures are marked by red dots. The BFGS method required 101 iterations to attain the stable states with 8496 individual beam energy evaluations and 0.256s of CPU time (13.389s with `ADTAGE0`). One can easily see the effect of rigid joints in 3 dimensions in the figures. The relative angles of all three beams meeting in a Frame-point are maintained during deformation.



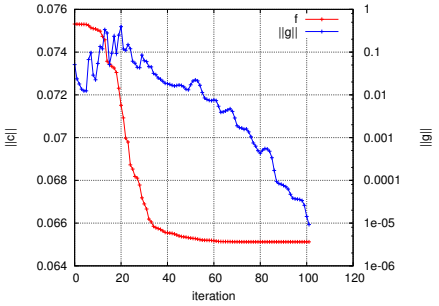
(a) undeformed



(b) load $= (0, -5, 1) \times 10^3$ N



(c) load $= (0, -5, -1) \times 10^3$ N



(d) iteration history

Figure 8.5: Pyramid with load at apex

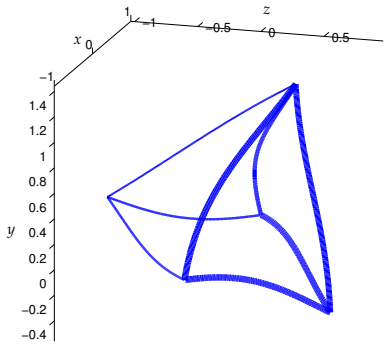
Evidently, ADTAGEO requires much more CPU time to do the same computations, which may be mostly ascribed to memory management during on the fly operations on the computational graph, especially vertex elimination that is done as soon as any variable goes out of scope. These operations on the computational graph is done each time the function is evaluated. The time required is directly proportional to the number of individual beam energy evaluations. It is also proportional to the number of free variables as we must access a partial derivative stored in some data structure for each free variable. Modulo these two factors, from the above mentioned timings, the CPU time requirement for ADTAGEO derivatives seems to be approximately between 3 to 5 times that of hard-coded derivative formulae as computed in chapter 7.

8.3 Design optimisation

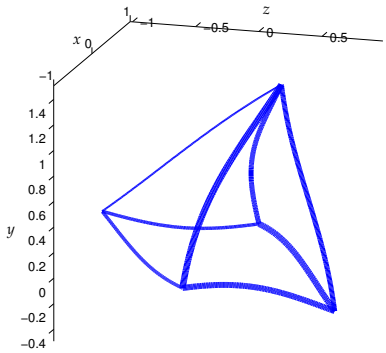
Using the total quasi-Newton optimisation software LRAMBO (chapter 3 we solve the constrained minimisation problem in (6.8) for some chosen parameter μ. The design variables are the areas of crossection of the beams.

Example 8.4. For the pyramid frame in Example 8.3 with $\mu = 2 \times 10^2$ J/kg, $\mu = 10^2$ J/kg and $\mu = 1$ J/kg solving (6.8) stiffens the compliant structure. The mass of the frame is changed by changing the crossection areas of the beams. Figure 8.6 shows the weakened and strengthened deformed structures. The thicker lines in the figure denote stronger beams but are not to scale. The mass of the original structure is 7.626 kg and a compliance of -5.983×10^3 J. The structure in Figure 8.6a and Figure 8.6b is stiffened by weakening some and strengthening other beams as required but in fact there is a decrease in the total mass due to a high μ. For Figure 8.6a mass is 5.516 kg and compliance is -6.057×10^3 J. For Figure 8.6b mass is 7.42 kg and compliance is -6.423×10^3 J. For the structure in Figure 8.6c mass is increased to 47.96 kg and compliance is -7.5×10^3 J. The respective design vectors change from all beams equal at 0.754 cm^2 to the following

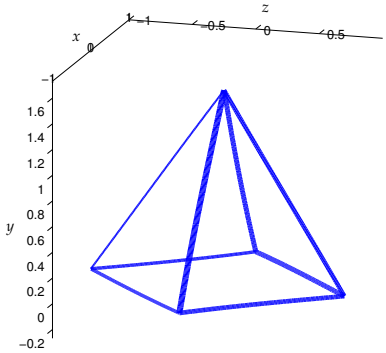
$$[0.013, 0.925, 1.099, 0.015, 0.002, 0.297, 0.877, 1.106] \text{ cm}^2 \quad \text{for Figure 8.6a}$$
$$[0.186, 1.304, 1.202, 0.212, 0.043, 0.715, 0.729, 1.469] \text{ cm}^2 \quad \text{for Figure 8.6b}$$
$$[2.514, 5.861, 4.918, 2.655, 1.578, 5.752, 4.405, 9.601] \text{ cm}^2 \quad \text{for Figure 8.6c}$$



(a) $\mu = 2 \times 10^2$ J/kg



(b) $\mu = 10^2$ J/kg



(c) $\mu = 1$ J/kg

Figure 8.6: Stiffened pyramid

The ordering of the beams is as follows:

1. $(0,0,-1) \leftrightarrow (1,0,0)$; 2. $(1,0,0) \leftrightarrow (0,0,1)$; 3. $(0,0,1) \leftrightarrow (-1,0,0)$;
4. $(-1,0,0) \leftrightarrow (0,0,-1)$; 5. $(0,0,-1) \leftrightarrow (0,1.5,0)$; 6. $(1,0,0) \leftrightarrow (0,1.5,0)$;
7. $(0,0,1) \leftrightarrow (0,1.5,0)$; 8. $(-1,0,0) \leftrightarrow (0,1.5,0)$.

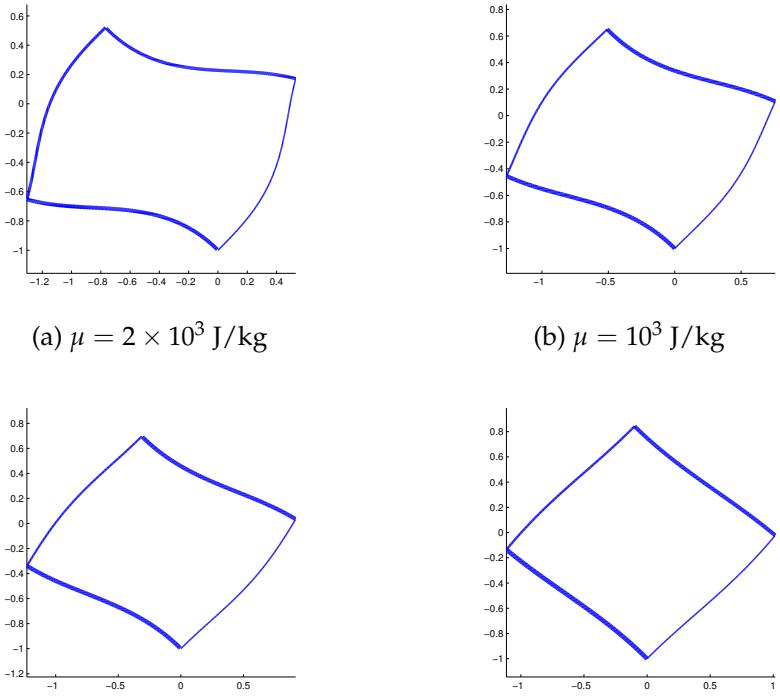
One can see the tendency that a large μ will lead to a decrease in mass by making some beams very thin. These thin beams will eventually be deleted at an even higher μ by setting their crossection area to be zero. Each run of the design optimisation took approximately 0.34s to 0.36s of CPU time to complete.

Example 8.5. Also for the rhombus in Example 8.2 we see the effect of changing μ from 2×10^3 J/kg to 1×10^2 J/kg in Figure 8.7. Stating with a mass of 3.352 kg and all beams of area 0.754 cm^2 the respective final mass and design vectors

are as follows

$[0.543, 0.844, 0.737, 1.071]$ cm^2; mass = 3.552kg; for Figure 8.7a;
$[0.631, 1.552, 0.668, 1.686]$ cm^2; mass = 5.044kg; for Figure 8.7b;
$[0.759, 2.063, 0.837, 2.202]$ cm^2; mass = 6.517kg; for Figure 8.7c;
$[1.086, 4.292, 1.473, 6.023]$ cm^2; mass = 14.313kg; for Figure 8.7d.

The ordering of the beams is anticlockwise starting from the lower right beam. The CPU times required to compute the design optimum were approximately 0.23s to 0.26s.



(a) $\mu = 2 \times 10^3$ J/kg (b) $\mu = 10^3$ J/kg

(c) $\mu = 5 \times 10^2$ J/kg (d) $\mu = 10^2$ J/kg

Figure 8.7: Stiffened rhombus

By making μ smaller we allow for more mass, and thus, smaller total energy of the frame. A large μ on the other hand tries to keep the mass small, and thus, the total energy of the frame is larger. Figure 8.8 shows this behaviour. The curve may be called a Pareto Front [KUNG et al., 1975] in terms of multi-objective and vector optimisation as one may choose any solution point on it depending on what compromise one wishes to make in terms of higher energy or higher mass.

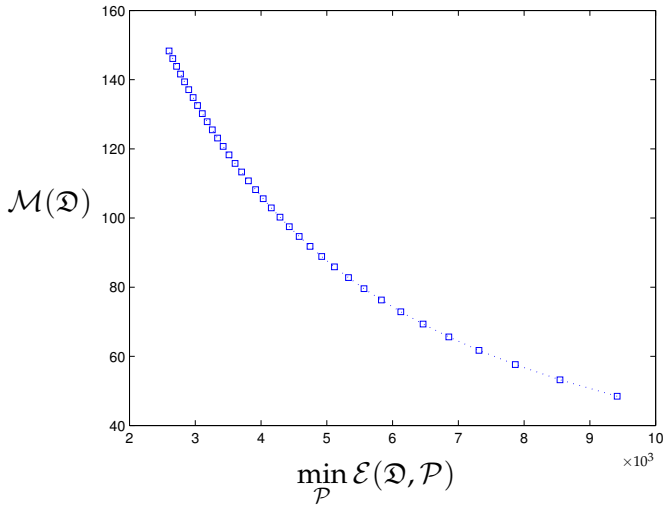


Figure 8.8: Variation of the total frame energy and its mass due to change in μ

8.4 Topological sensitivities

For any given stable state and design the topological sensitivity for adding a beam between any two points on different beams may be computed as described in subsection 6.2.2.

Example 8.6. Figure 8.9 shows a representation of the topological sensitivity for the pyramid in Example 8.4 with $\mu = 1$ J/kg. Each square block shows to sensitivity for adding a zero sized beam between two of the currently present beams starting from the top left. The ordering of the beams is the same as in Example 8.4. The green colour represents a small sensitivity value. The gradual colour change from green to yellow to red represents the increase in the sensitivity values. The deep red areas are where the sensitivity is extremely large due to very strong tension, and thus, large elongation. It should be noted that adding a beam between points that show a large sensitivity and then running the optimal sizing algorithm again shall result in a lower compliance. It required 23104 virtual beam energy computations to compute the sensitivity chart and a CPU time of 2.66s.

Example 8.7. We have a pre-design for a small bridge with only the base form that may be extended using the results of the topological sensitivities. Figure 8.10a shows the undeformed planar case with the load applied at the centre in the negative y direction and the two lower endpoints marked in red being fixed in position as well as orientation. The resulting deformed shape is in Figure 8.10b along with the ordering of the beams. The topological sensitivities are

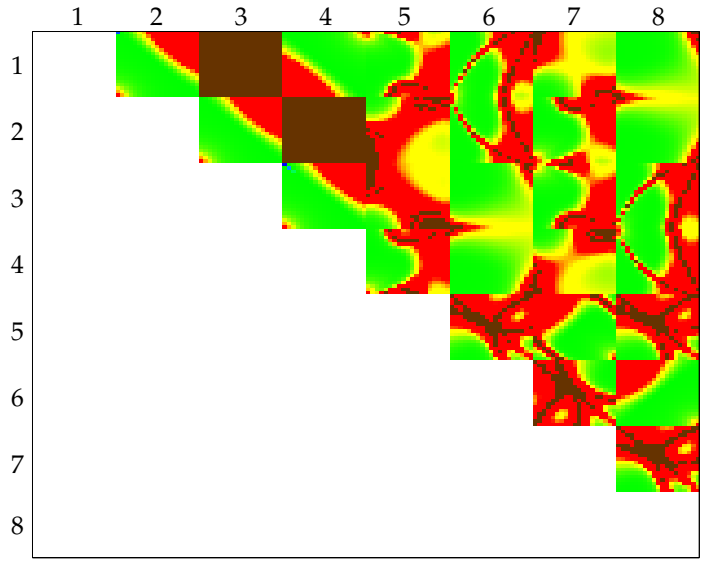


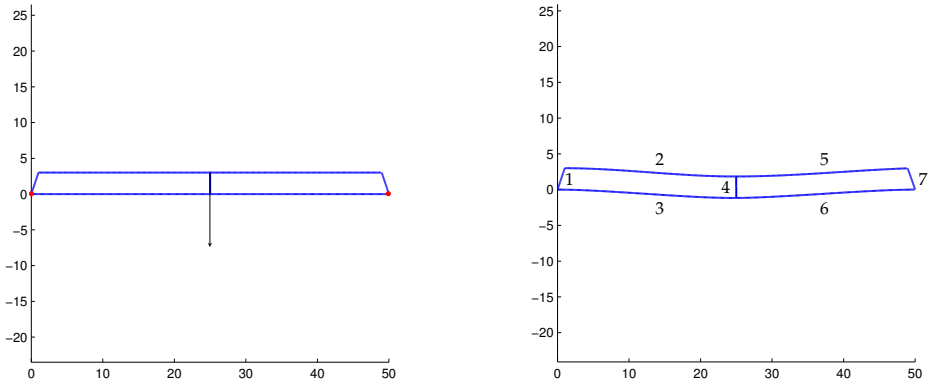
Figure 8.9: Topological sensitivity for pyramid

plotted in Figure 8.10c. Noticeably the connection of any two points between beams 1 and 4, 4 and 7 or 1 and 7 does not have any significant improvement in the design. Also, although one may compute the sensitivity of connections between beams 1 and 5 or 1 and 6 and analogously 2 and 7 or 3 and 7, these are physically infeasible due to material crossing each other. The computation of the sensitivity chart required 17689 virtual beam energy evaluations and a CPU time of 1.844s.

The topological sensitivities computed are, therefore, a tool that facilitates further design optimisation. However, physical constraints need to be considered before relying on the values. The computation of the sensitivity chart is quite fast for the examples considered and this chart may be used for interactive design optimisation.

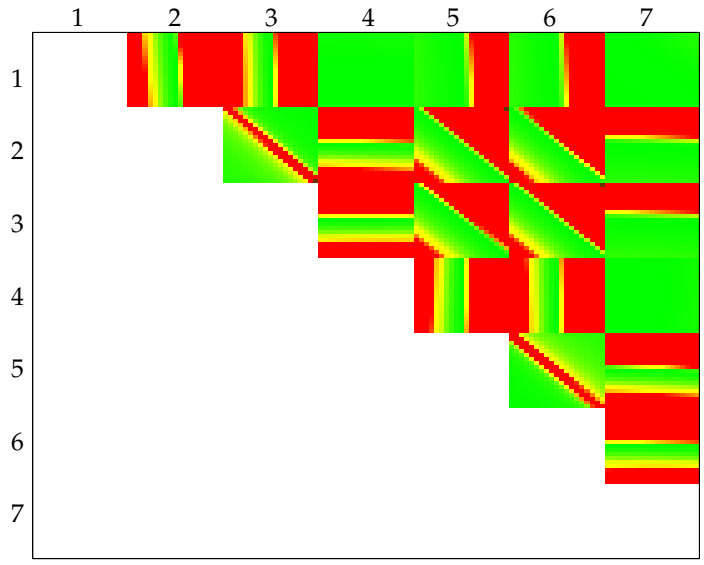
8.5 A larger computation

Example 8.8 (The leaning tower of Eiffel)**.** Let us look at a hypothetical large scale computation simply to see if the algorithm perfoms also on a larger computation than a handful of variables. We consider a planar version of the Eiffel tower as shown in Figure 8.11a under gravity that is acting inclined, as if the tower were leaning sideways at an angle of 20°. The problem has 3208 variables and 1731 beams. All the 8 base points are fixed both in position and ori-



(a) undeformed

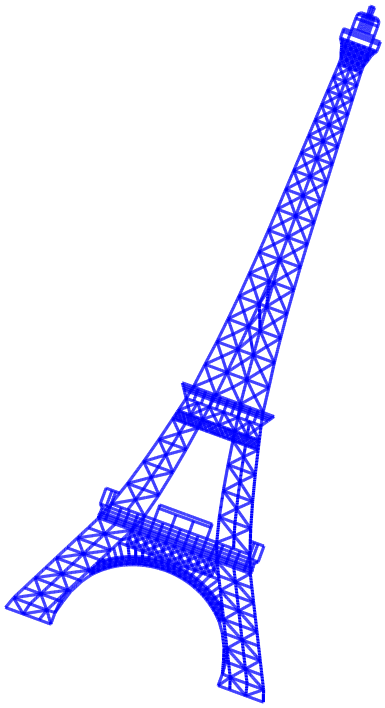
(b) deformed



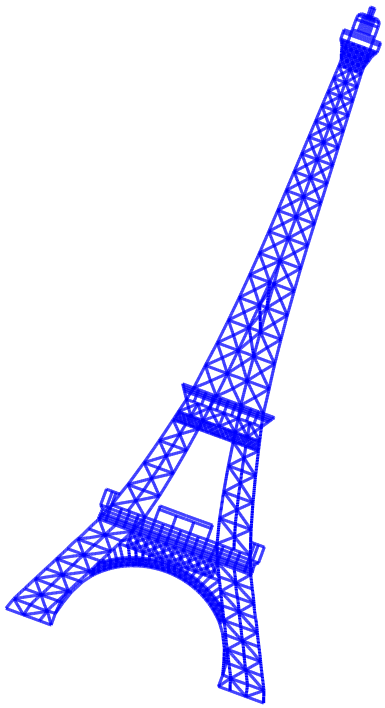
(c) topological sensitivities

Figure 8.10: Pre-design for a bridge

entation. The computation requires 423 optimisation iterations with 7941828 individual beam energy evaluations and a total CPU time of 11m3.529s. As Figure 8.11b shows the distortion due to gravity is not large enough to be noticeable on the scale of the figure. The total height of the tower is 300m. The 2-norm of the discrepancy in the state vector between the two states is 2.422m and the maximal discrepancy component (infinity norm) is 0.140m.



(a) undeformed



(b) deformed under inclined gravity

Figure 8.11: Eiffel tower example

Chapter 9

Concluding remarks

A discretisation free method for modelling and optimisation of structures consisting of flexible beams and rigid joints has been presented in the previous chapters. The discussed model considers the elongation, compression as well as the bending of the beams comprising a flexible frame. The optimisation task is formulated in terms of the stored energy and the external work. The design optimisation task is a saddle point problem as discussed in chapter 6.

Although the basic idea of the total elastic energy is simple, the dependence of the bending energy on the axial compression or elongation force P is complicated and its determination from a modified Hooke's Law using the internal safeguarded Newton method turns out to be the most critical issue. This is also the reason for the derivative computation being nested and unsuitable for classical AD tools that are unable to compute such nested derivatives. The solution process also quite sensitive due to the presence of a singularity that may cause a degeneracy in certain cases. The variation of P or the scaled and dimensionless W with the state has been shown to be at least Lipschitz in the degenerate case.

The current software implementation is a numerical validation tool and a proof of concept. The numerical results in chapter 8 are promising; it is however, admittedly, by far not the most efficient implementation of the model presented. For moderately sized problems one can expect interactive design optimisation via the topological sensitivities. The algorithm is however scalable for large problems too, and further speed-up may be obtained with a more efficient implementation that exploits the inherent parallelism in the individual beam energy computations.

Currently the model presented in chapter 4 considers only axially symmetric beams and does not take into account the effect of axial torsion. The model can be further extended to include energy due to torsion, using the quaternions

that determine the orientation of the ends or the beams, and also beams that are not axially symmetric by introducing more design variables. The design variables in non symmetric case, however, cannot be regarded as dual variables as in section 6.2. We must solve the saddle point problem (6.1) and topological sensitivity computation similar to subsection 6.2.2 would require further differentiation with respect to the new design variables. A more efficient and faster implementation of `ADTAGEO` that also computes second derivatives will be a step forward to achieve this as one would not have to precompute the derivatives as in chapter 7. This precomputation is tedious and makes the incorporation of non symmetric beams a daunting task at the moment. A low rank update based optimiser as that of GRIEWANK et al. [2004–2010] that can maintain quasi-definiteness property of the Hessian in order to solve saddle point problems as in (6.6) is another yet missing tool towards this end.

Appendix A

Properties of $\widetilde{L}_{\text{odd}\,|\,\text{even}}$

Let us look at the expressions for $\widetilde{L}_{\text{odd}\,|\,\text{even}}$ as defined in section 4.4 on the region $-\infty < W < \pi^2$, especially the closed forms in (4.12) in order to prove Proposition 4.1.

We have

$$\widetilde{L}_{\text{even}} = \begin{cases} \frac{1}{8}\left[\frac{2u-\sin 2u}{u(1-\cos 2u)}\right] & \text{if } W > 0\,, \quad u = \sqrt{W} \\ \frac{1}{12} & \text{if } W = 0 \\ \frac{1}{8}\left[\frac{\sinh 2u-2u}{u(\cosh 2u-1)}\right] & \text{if } W < 0\,, \quad u = \sqrt{-W} \end{cases}$$

From the above it is quite clear that $L_{\text{even}} > 0$ for $-\infty < W < \pi^2$ and

$$\lim_{W\to-\infty} L_{\text{even}} = \lim_{u\to\infty} \frac{\sinh 2u - 2u}{u(\cosh 2u - 1)} = 0$$

The derivative of L_{even} w.r.t. W is the following

$$\frac{\mathrm{d}\widetilde{L}_{\text{even}}}{\mathrm{d}W} = \begin{cases} \frac{1}{2}\frac{-v\cos v-\sin v\cos v+(1-v^2)\sin v+v}{v^3(1-\cos v)^2} & \text{if } W > 0, \quad v = 2\sqrt{W} \\ \frac{1}{90} & \text{if } W = 0 \\ \frac{1}{2}\frac{\sinh v\cosh v+v\cosh v-(1+v^2)\sinh v-v}{v^3(\cosh v-1)^2} & \text{if } W < 0, \quad v = 2\sqrt{-W} \end{cases} \tag{A.1}$$

In the respective denominators we have $2(1-\cos v)^2 > 0$ and $2(\cosh v - 1)^2 > 0$, we now expand remaining factor of the trigonometric and hyperbolic expressions into their power series form

$$\sum_{j=0}^{\infty} \frac{(-1)^j 4}{(2j+3)!}(2^{2j} - j^2 - 2j - 1)v^{2j} \quad \text{for } v = 2\sqrt{W}\,,\ W > 0$$

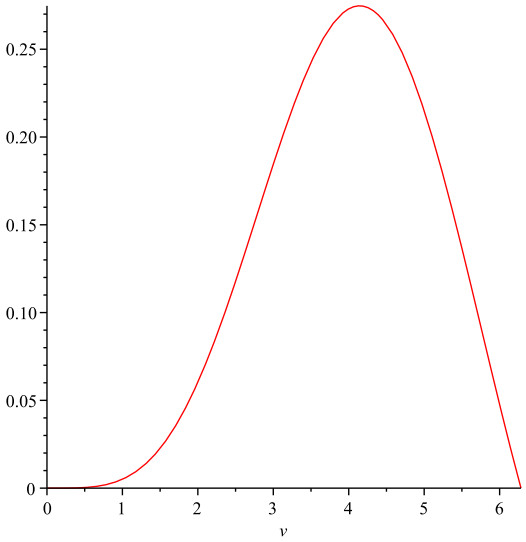


Figure A.1: Trigonometric numerator in $\frac{\mathrm{d}\widetilde{L}_{\mathrm{even}}}{\mathrm{d}W}$ for $0 < v < 2\pi$.

$$\sum_{j=0}^{\infty} \frac{4}{(2j+3)!}(2^{2j} - j^2 - 2j - 1)v^{2j} \quad \text{for } v = 2\sqrt{-W},\ W < 0$$

Both together can be written in terms of W as

$$\sum_{j=0}^{\infty} \frac{(-1)^j 2^{2j+2}}{(2j+3)!}(2^{2j} - j^2 - 2j - 1)W^j$$

It is easy to see that for the hyperbolic series all the coefficients are positive for $j \geqslant 2$ and rapidly declining. For the trigonometric expression we use the first two nontrivial terms in the Leibniz criterion for alternating series [KNOPP, 1956] and get

$$\text{expr} > \frac{1}{3780}v^4(21 - 2v^2) \quad \Longrightarrow \quad \text{expr} > 0 \text{ for } 0 < v < \sqrt{\frac{21}{2}} \approx 3.24037$$

The trigonometric expression is shown in Figure A.1 for $0 < v < 2\pi$. One can conclude that the numerators of the expressions in (A.1) are also positive for $-\infty < W < \pi^2$ and therefore the derivative $\dfrac{\mathrm{d}\widetilde{L}_{\mathrm{even}}}{\mathrm{d}W} > 0$. The above expressions also yield

$$\lim_{W\to-\infty} \frac{\mathrm{d}\widetilde{L}_{\mathrm{even}}}{\mathrm{d}W} = \lim_{v\to\infty} \frac{\sinh v \cosh v + v \cosh v - (1+v^2)\sinh v - v}{v^3(\cosh v - 1)^2} = 0$$

since both the denominators $(\cosh v - 1)^2$ and $(1 - \cos v)^2$ dominate respectively. The common power series expansion for the denominators in terms of

W is

$$\sum_{j=0}^{\infty} \frac{(-1)^{j+1} 2^{2j+1}}{(2j+2)!} (2^{2j+2} - 4) W^{j+1}$$

Both numerator and denominator have the leading order W^2, i.e. v^4, and the respective coefficients in the numerator are all much smaller than the ones in the denominator.

Once again differentiating (A.1) w.r.t. W yields

$$\frac{\mathrm{d}^2 \widetilde{L}_{\text{even}}}{\mathrm{d}W^2} = \begin{cases} \frac{3\sin v \cos v + (3v + v^3)\cos v - 3\sin v - 3v + 2v^3}{v^5 (1 - \cos v)^2} & \text{if } W > 0,\ v = 2\sqrt{W} \\ \frac{1}{315} & \text{if } W = 0 \\ \frac{3\sinh v \cosh v + (3v - v^3)\cosh v - 3\sinh v - 3v - 2v^3}{v^5 (\cosh v - 1)^2} & \text{if } W < 0,\ v = 2\sqrt{-W} \end{cases} \tag{A.2}$$

As before we have in the respective denominator $(1 - \cos v)^2 > 0$ and $(\cosh v - 1)^2 > 0$ and we expand the remaining factor of the trigonometric and hyperbolic expression into their power series form

$$\sum_{j=0}^{\infty} \frac{(-1)^j}{(2j+5)!} [3 \cdot 2^{2j+4} - 2^3 (j^3 + 6j^2 + 11j + 6)] v^{2j} \quad \text{for } v = 2\sqrt{W},\ W > 0$$

$$\sum_{j=0}^{\infty} \frac{1}{(2j+5)!} [3 \cdot 2^{2j+4} - 2^3 (j^3 + 6j^2 + 11j + 6)] v^{2j} \quad \text{for } v = 2\sqrt{-W},\ W < 0$$

Both these series can be written together in terms of W as

$$\sum_{j=0}^{\infty} \frac{(-1)^j 2^{2j}}{(2j+5)!} [3 \cdot 2^{2j+4} - 2^3 (j^3 + 6j^2 + 11j + 6)] W^j$$

One can again easily see that for the hyperbolic series the coefficients are all positive for $j \geqslant 2$ and rapidly declining. For the trigonometric expression we use the first two nontrivial terms in the Leibniz criterion for alternating series [KNOPP, 1956] and get

$$\text{expr} > \frac{1}{18900} v^4 (15 - v^2) \quad \Longrightarrow \quad \text{expr} > 0 \text{ for } 0 < v < \sqrt{15} \approx 3.872983$$

The trigonometric expression is shown in Figure A.2 for $0 < v < 2\pi$. One can conclude that the numerator is also positive for $-\infty < W < \pi^2$ and therefore

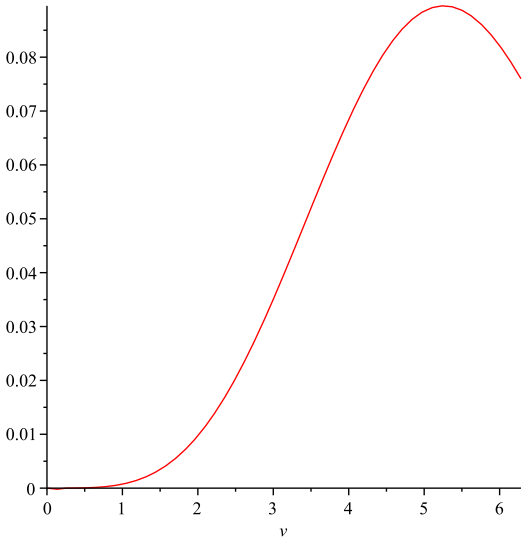


Figure A.2: Trigonometric numerator in $\frac{\mathrm{d}^2\widetilde{L}_{\mathrm{even}}}{\mathrm{d}W^2}$ for $0 < v < 2\pi$.

the derivative in (A.2) is positive. Also

$$\begin{aligned}\lim_{W\to-\infty}\frac{\mathrm{d}^2\widetilde{L}_{\mathrm{even}}}{\mathrm{d}W^2} = \\ \lim_{v\to\infty}\frac{3\sinh v\cosh v + 3v\cosh v - v^3\cosh v - 3\sinh v - 3v - 2v^3}{v^5(\cosh v - 1)^2} \\ = 0\end{aligned}$$

due to the dominating power series of the denominator. The leading order of the numerator series is again W^2 or v^4 as also of the denominator and the respective coefficients in the numerator are much smaller than the ones in the denominator.

For $\widetilde{L}_{\mathrm{odd}}$ we have from (4.12)

$$\widetilde{L}_{\mathrm{odd}} = \begin{cases} \frac{1}{8}\left[\frac{u^2-2\sin^2 u+u\sin u\cos u}{(u\cos u-\sin u)^2}\right] & \text{if } W>0\,,\ u=\sqrt{W} \\ \frac{1}{20} & \text{if } W=0 \\ \frac{1}{8}\left[\frac{u^2-2\sinh^2 u+u\sinh u\cosh u}{(u\cosh u-\sinh u)^2}\right] & \text{if } W<0\,,\ u=\sqrt{-W} \end{cases}$$

Clearly one can see that $\widetilde{L}_{\mathrm{odd}} > 0$ for $-\infty < W < \pi^2$ and

$$\lim_{W\to-\infty}\widetilde{L}_{\mathrm{odd}} = \lim_{u\to\infty}\frac{u^2 - 2\sinh^2 u + u\sinh u\cosh u}{(u\cosh u - \sinh u)^2} = 0$$

Differentiating once yields for

$$\frac{\mathrm{d}\widetilde{L}_{\mathrm{odd}}}{\mathrm{d}W} = \begin{cases} \frac{1}{256}\frac{(12-v^2)\sin 2v+(2v^4-34v^2-24)\sin v-8v\cos 2v-10(4v-v^3)\cos v+2v^3+48v}{v(u\cos u-\sin u)^4} \\ \qquad \text{if } W>0\,, \quad u=\sqrt{W}\,,\ v=2u \\ \frac{1}{350} \qquad \text{if } W=0 \\ \frac{1}{256}\frac{(12+v^2)\sinh 2v+(2v^4+34v^2-24)\sinh v-8v\cosh 2v-10(4v+v^3)\cosh v-2v^3+48v}{v(u\cosh u-\sinh u)^4} \\ \qquad \text{if } W<0\,, \quad u=\sqrt{-W}\,,\ v=2u \end{cases} \tag{A.3}$$

We proceed analogous to $\widetilde{L}_{\mathrm{even}}$, here in the respective denominator we have $256(u\cos u-\sin u)^4>0$ and $256(u\cosh u-\sinh u)^4>0$, and we expand the remaining factors into their respective power series form.

$$\sum_{j=0}^{\infty}\frac{(-1)^j}{(2j+5)!}\{2^{2j+4}[2j^2-7j-6]+2^4(2j^4+9j^3+14j^2+13j+6)\}v^{2j+4}$$

$$\text{for } W>0 \ v=2\sqrt{W}$$

$$\sum_{j=0}^{\infty}\frac{1}{(2j+5)!}\{2^{2j+4}[2j^2-7j-6]+2^4(2j^4+9j^3+14j^2+13j+6)\}v^{2j+4}$$

$$\text{for } W<0 \ v=2\sqrt{-W}$$

Both series together can be written in terms of W as

$$\sum_{j=0}^{\infty}\frac{2^{2j+4}(-1)^j}{(2j+5)!}\{2^{2j+4}[2j^2-7j-6]+2^4(2j^4+9j^3+14j^2+13j+6)\}W^{j+2}$$

Note that all the coefficients in the hyperbolic series are positive for $j\geqslant 4$ and rapidly declining. For the trigonometric expression we use the first two nontrivial terms in the Leibniz criterion for alternating series [KNOPP, 1956] and get

$$\text{expr}>\frac{1}{6804000}v^{12}(15-v^2) \quad\Longrightarrow\quad \text{expr}>0 \text{ for } 0<v<\sqrt{15}\approx 3.872983$$

The trigonometric expression is shown in Figure A.3 for $0<v<2\pi$. Thus, once again, the numerator is positive for $-\infty<W<\pi^2$ and one conclude that

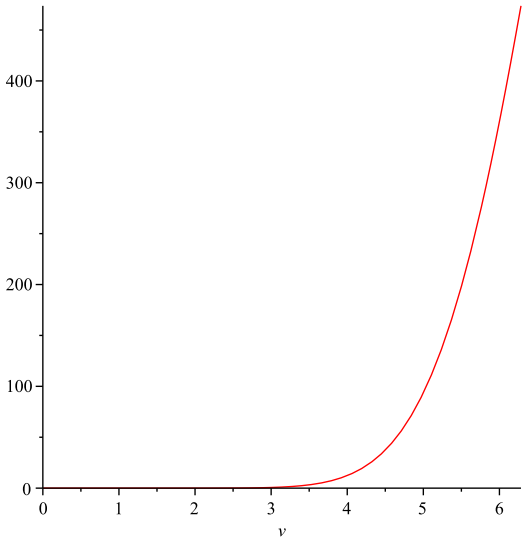


Figure A.3: Trigonometric numerator in $\frac{\mathrm{d}\widetilde{L}_{\mathrm{odd}}}{\mathrm{d}W}$ for $0 < v < 2\pi$.

the derivative in (A.3) is positive. Also

$$\lim_{W\to-\infty}\frac{\mathrm{d}\widetilde{L}_{\mathrm{odd}}}{\mathrm{d}W} = \lim_{\substack{u\to\infty\\ v=2u}}\frac{(12+v^2)\sinh 2v+(2v^4+34v^2-6)\sinh v-8v\cosh 2v-10(4v+v^3)\cosh v-2v^3+48v}{v(u\cosh u-\sinh u)^4} = 0$$

as $(u\cos u-\sin u)^4$ and $(u\cosh u-\sinh u)^4$ dominate respectively. The respective denominators $(u\cos u-\sin u)^4$ and $(u\cosh u-\sinh u)^4$ can both be written jointly in terms of W as the series

$$\sum_{j=0}^{\infty}\frac{2^{2j+6}(-1)^{j+1}}{(2j+6)!}\{2^{2j+5}[4j^4+4j^3-25j^3-21j+10] + 2^5[4j^4+28j^3+59j^2+31j-10]\}W^{j+3}$$

The leading order in both the numerator and the denominator is W^6, i.e. v^{12}, and the respective coefficients in the numerator are much smaller than the ones in the denominator.

Differentiating (A.3) once more w.r.t. W yields

$$\frac{\mathrm{d}^2\widetilde{L}_{\mathrm{odd}}}{\mathrm{d}W^2} = \begin{cases} \frac{1}{128}\frac{(3v^2-12)\sin 2v+(30v^2+8v^4+24)\sin v+24v\cos 2v+(-2v^5+26v^3-24v)\cos v+4v^5-38v^3}{v^3(u\cos u-\sin u)^4} \\ \qquad \text{if } W>0\,, \quad u=\sqrt{W}\,,\ v=2u \\ \frac{1}{2625} \qquad \text{if } W=0 \\ \frac{1}{128}\frac{(3v^2+12)\sinh 2v+(30v^2-8v^4-24)\sinh v-24v\cosh 2v+(2v^5+26v^3+24v)\cosh v-4v^5-38v^3}{v^3(u\cosh u-\sinh u)^4} \\ \qquad \text{if } W<0\,, \quad u=\sqrt{-W}\,,\ v=2u \end{cases} \tag{A.4}$$

Once again we have the respective denominators $128(u\cos u-\sin u)^4>0$ and $(128(u\cosh u-\sinh u)^4>0$, and we expand the remaining factors into their respective power series form

$$\sum_{j=0}^{\infty}\frac{(-1)^j 2^5}{(2j+7)!}\{2^{2j}[12j^2-18j-162] + 2j^5+21j^4+85j^3+177j^2+225j+162\}v^{2j+4} \qquad \text{for } W>0\,,\ v=2\sqrt{W}$$

$$\sum_{j=0}^{\infty}\frac{2^5}{(2j+7)!}\{2^{2j}[12j^2-18j-162] + 2j^5+21j^4+85j^3+177j^2+225j+162\}v^{2j+4} \qquad \text{for } W<0\,,\ v=2\sqrt{-W}$$

or in terms of W itself

$$\sum_{j=0}^{\infty}\frac{2^{2j+9}(-1)^j}{(2j+7)!}\{2^{2j}[12j^2-18j-162] + 2j^5+21j^4+85j^3+177j^2+225j+162\}W^{j+2}$$

One can once again easily see that the coefficients in the hyperbolic series are all positive for $j\geqslant 4$ and decline rapidly. For the trigonometric expression we use the first two nontrivial terms in the Leibniz criterion for alternating series

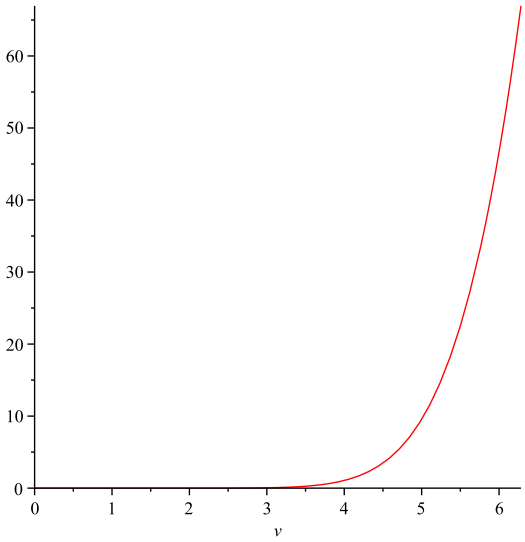


Figure A.4: Trigonometric numerator in $\frac{\mathrm{d}^2\widetilde{L}_{\mathrm{odd}}}{\mathrm{d}W^2}$ for $0 < v < 2\pi$.

[KNOPP, 1956] and get

$$\begin{aligned} \text{expr} &> \frac{1}{523908000} v^{12}(77 - 4v^2) \\ \Longrightarrow \text{expr} &> 0 \text{ for } 0 < v < \frac{\sqrt{77}}{2} \approx 4.387482 \end{aligned}$$

The trigonometric expression is shown in Figure A.4 for $0 < v < 2\pi$. Thus, once again one concludes that the derivative in (A.4) is positive for $-\infty < W < \pi^2$.

Also

$$\begin{aligned} &\lim_{W\to-\infty} \frac{\mathrm{d}^2\widetilde{L}_{\mathrm{odd}}}{\mathrm{d}W^2} \\ &= \lim_{\substack{u\to\infty \\ v=2u}} \frac{(3v^2+12)\sinh 2v+(30v^2-8v^4-24)\sinh v-24v\cosh 2v+(2v^5+26v^3+24v)\cosh v-4v^5-38v^3}{v^3(u\cosh u-\sinh u)^4} \\ &= 0 \end{aligned}$$

as the denominator dominates. The leading order in the numerator is W^6, i.e. v^{12} and the respective coefficients in the numerator are much smaller than those in the denominator.

This shows that Proposition 4.1 is indeed true.

Addendum

According to the recommendation of the doctoral committee I submit the following addendum to subsection 6.2.2 of my dissertation.

Tue 19 Oct, 2010 Kshitij Kulshreshtha

Topological sensitivity

The topological derivative as defined in the works of CÉA, GARREAU, GUILLAUME & MASMOUDI [2000]; GARREAU, GUILLAUME & MASMOUDI [2000]; MASMOUDI [1998]; SOKOLOWSKI & ZOCHOWSKI [1999, 2002] is the result of the limiting process of introducing an infinitesimal ball into a continuum structure with different material properties. In this scenario the boundary conditions acting on the surface of the ball, i. e. at the interface where the material properties change, play a very crucial role in the computation of this derivative. The limiting process, in which the infinitesimal ball tends to zero, changes the interface where these boundary conditions must act too. Dirichlet boundary conditions on the deformation are, in general, mathematically tricky to handle in such a scenario and may result in extra terms in the topological derivative that are not necessarily proportional to the volume variation.

In the case of a frame, we reuse the term *topological sensitivity* because we also consider an infinitesimal modification that changes the structure of the frame design rather than just its sizing. When considering the topological sensitivity for the addition of an extra beam between two points on the frame, as defined in subsection 6.2.2, page 56, at the first glance one sees such Dirichlet boundary conditions at the new joints due the incidence and orientation constraints. However in this case the design variable, i. e. the variable of differentiation, unlike the infinitesimal ball, is not the volume of this extra beam but only its area of cross-section. The length of the beam is a constant which is chosen according to the state of the undeformed beam to keep the undeformed elastic energy trivial. Thus the incidence constraints are always satisfied and appear

only as constants in the limiting process. The relative orientation at the joint is also constant for the computation of this topological sensitivity as the deformed shape close to the limit $d_i \to 0$ for the new beam i remains the same as it would be without this beam. The term $e_i(s) - \frac{\rho_i}{\mu}$ signifies the marginal benefit of introducing this beam into the structure in terms of its energy density relative to its mass density, and answers the question whether introducing this beam into the structure may lead to a better solution to our design problem for the given parameter μ.

Bibliography

ACHTZIGER, W. (1993). *Optimierung von einfach und mehrfach belasteten Stabwerken*. *Bayreuther Mathematische Schriften*, **Heft 46**.

ACHTZIGER, W. (1996). *Truss topology optimization including bar properties different for tension and compression*. *Structural Optimization*, **12** (1) 63–73.

ANTMAN, S. S. (2005). *Nonlinear Problems of Elasticity*, vol. 107 of *Applied Mathematical Sciences*. Springer, 2nd edn.

ARFKEN, G. (1985). *Mathematical Methods for Physicists*, chap. SS5.9 "Bernoulli Numbers, Euler-Maclaurin Formula", pp. 327–338. Academic Press.

ARNOLD, V. I. (1989). *Mathematical Methods of Classical Mechanics*. Springer, 2nd edn.

BAŽANT, Z. P. & CEDOLIN, L. (1991). *Stability of Structures*. Oxford University Press.

BELL, B. M. (2003–2010). *CppAD: A Package for C++ Algorithmic Differentiation*. Homepage. `http://www.coin-or.org/CppAD/`.

BENDSØE, M. P. & SIGMUND, O. (2003). *Topology Optimization*. Springer, 2nd edn.

BERNOULLI, J. (1713). *Ars Conjectandi: Wahscheinlichkeitsrechnung*, chap. Summae Potestatum: Summe der Potenzen. Deutsch Verlag (originally Thurnisiorum). Translated from Latin by Robert Haußner (1899).

BISCHOF, C., CARLE, A., CORLISS, G., GRIEWANK, A. & HOVLAND, P. (1992). *ADIFOR - Generating Derivative Codes from Fortran Programs*. *Scientific Programming*, **No. 1** 1–29.

BISCHOF, C., CARLE, A., KHADEMI, P. & MAUER, A. (1994). *The ADIFOR 2.0 System for the Automatic Differentiation of Fortran 77 Programs*. Tech. Rep. ANL/MCS-P481-1194, Arragone National Laboratory.

BISCHOF, C. & ROH, L. (1997). *ADIC: An Extensible Automatic Differentiation Tool for ANSI-C*. Tech. Rep. ANL/MCS-P626-1196, Arragone National Laboratory.

BOSSE, T., SCHLOSSHAUER, V. & GRIEWANK, A. (2009). *On Hessian- and Jacobian-free SQP methods - a total quasi-Newton scheme with compact storage*. In *4th. Belgian-French-German Conference on Optimization*. Springer, Leuven. Submitted.

BOSSE, T. F. (2009). *A derivative-matrix-free NLP Solver without explicit nullspace representation*. Diplomarbeit, Humboldt-Universtät zu Berlin.

DE BRUIJN, N. G. (1981). *Asymptotic Methods in Analysis*. Dover, New York.

CONN, A. R., GOULD, N. I. M. & TOINT, P. L. (2000). *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM.

CONN, A. R., GOULD, N. I. M. & TOINT, P. L. (2002). *The SIF Reference Document*. Unpublished.
URL `http://www.numerical.rl.ac.uk/lancelot/sif/sifhtml.html`

CORLESS, R. M., GONNET, G. H., HARE, D. E. G., JEFFREY, D. J. & KNUTH, D. E. (1996). *On the Lambert W function*. *Adv. Computational Maths.*, **5** 329–359.

CÉA, J., GARREAU, S., GUILLAUME, P. & MASMOUDI, M. (2000). *The shape and topological optimizations connection*. *Comput. Method. Appl. M.*, **188** (4) 713 – 726.

DEMPE, S. (2002). *Foundations of Bilevel Programming*. Kluwer Academic Press, Dordrecht.

DORN, W., GOMORY, R. & GREENBERG, M. (1964). *Automatic design of optimal structures*. *J. de Mecanique*, **3** 25–52.

DUYSINX, P. & BENDSØE, M. P. (1998). *Topology optimization of continuum structures with local stress constraints*. *International Journal for Numerical Methods in Engineering*, **43** (8) 1453–1478.

EVANS, D. J. & MURAD, S. (1977). *Singularity free algorithm for molecular dynamics simulations of rigid polyatomics*. *Molecular Physics*, **34** (2) 327–331.

FLERON, P. (1964). *The minimum weight of trusses*. *Bygningsstatiske Meddelelser*, **35** 81–96.

GARREAU, S., GUILLAUME, P. & MASMOUDI, M. (2000). *The Topological Asymptotic for PDE Systems: The Elasticity Case. SIAM J. Control Optim.*, **39** (6) 1756–1778.

GIERING, R. (1997). *Tangent Linear and Adjoint Model Compiler, Users Manual*. Center for Global Change Sciences, Department of Earth, Atmospheric, and Planetary Science, MIT, Cambridge, MA. Unpublished.
URL `http://puddle.mit.edu/~ralf/tamc`

GIERING, R. & KAMINSKI, T. (1998). *Recipes for Adjoint Code Construction. ACM Transactions on Mathematical Software*, **24** (4) 437–474.

GILL, P. E., SAUNDERS, M. A. & SHINNERL, J. R. (1996). *On the stability of cholesky factorization for symmetric quasidefinite systems. SIAM J. Matrix Anal. Appl.*, **17** (1) 35–46.

GOLDSTEIN, H. (1973). *Classical Mechanics*, chap. 4. Addison-Wesley, Reading, Mass.

GRIEWANK, A., JUEDES, D., MITEV, H., UTKE, J., VOGEL, O. & WALTHER, A. (1996). *ADOL-C: A Package for the Automatic Differentiation of Algorithms written in C/C++. ACM Transations on Mathematical Software*, **22** (2) 131–167. Algor. 755.

GRIEWANK, A., KÖRKEL, S., KULSHRESHTHA, K., BOSSE, T., EICHSTÄDT, S., SCHLOSSHAUER, V. & KORZEC, M. (2004–2010). *General purpose, linearly invariant algorithm for large-scale nonlinear programming*. DFG Research Center MATHEON, Project C12.
URL `http://www.math.hu-berlin.de/~griewank/C12/`

GRIEWANK, A. & NAUMANN, U. (2002). *Accumulating Jacobians by vertex, edge, and face elimination*. In 6^e *Colloque Africain sur la Recherche en Informatique*. INRIA.

GRIEWANK, A. & WALTHER, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2nd edn.

GRIEWANK, A., WALTHER, A. & KORZEC, M. (2007a). *Maintaining factorized KKT Systems subject to Rank-one Updates of Hessians and Jacobians. Optimization Methods and Software*, **22** (2) 279–295.

GRIEWANK, A., WALTHER, A. & KORZEC, M. (2007b). *Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians*. *Optimization Methods and Software*, **22** (2) 279–295.

GRIEWANK, A. O., MARKEY, B. R. & EVANS, D. J. (1979). *Singularity-free static lattice energy minimization*. *J. Chem. Phys.*, **71** (8) 3449–3454.

HASCOET, L. (2004). *TAPENADE: a tool for Automatic Differentiation of programs*. In *Proceedings of the ECCOMAS conference*. Jyväskylä, Finnland.

HEUSER, H. (2004). *Gewöhnliche Differentialgleichungen: Einführung in Lehre und Gebrauch*, chap. 27. Teubner, 4th edn.

KNOPP, K. (1956). *Infinite Series and Sequences*. Dover. ISBN 0-486-60153-6.

KNOPP, K. (1996). *Theory of Functions Part I and II, Two Volumes Bound as One, Part 1*, chap. 10, "The Laurent Expansion", pp. 117–122. Dover.

KREITERLING, S. (2007). *Effiziente Schrittweitenbestimmung für Optimierungsprobleme mit negativer Krümmung*. Diplomarbeit, Humboldt Universität zu Berlin.

KUNG, H. T., LUCCIO, F. & PREPARATA, F. P. (1975). *On finding the maxima of a set of vectors. Journal of the ACM*, **22** (4) 469—-476.

LANDAU, L. D. & LIFSCHITZ, E. M. (1986). *Theory of Elasticity*, vol. 7 of *Course of Theoretical Phyiscs*. Butterworth Heinemann, Boston, MA, 3rd edn. Translated from Russian by J.B. Sykes and W.H. Reid.

LOU, Z.-Q., J.-S., P. & RALPH, D. (1996a). *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press.

LOU, Z.-Q., PANG, J.-S., RALPH, D. & WU, S.-Q. (1996b). *Exact penalization and stationarity conditions of mathematical programs with equilibrium constraints*. *Mathematical Programming*, **75** 19–76.

LOVE, A. E. H. (1944). *A treatise on the mathematical theory of elasticity*. Dover Publications, 4th edn.

MASMOUDI, M. (1998). *A synthetic presentation of shape and topological optimization*. In *Proceedings of the Inverse Problems, Control and Shape Optimization, PICOF '98*.

MICHELL, A. G. M. (1904). *The limit of economy of material in frame structures*. *Philosophical Magazine*, **8** (6) 589–597.

NAUMANN, U. (2008). *Optimal Jacobian accumulation is NP-complete*. *Math. Prog.*, **112** (2) 427–441.

NAUMANN, U. & HU, Y. (2008). *Optimal vertex elimination in single-expression-use graphs*. *ACM Transactions on Mathematical Software*, **35** (1) 1–20.

NOCEDAL, J. & WRIGHT, S. J. (1999). *Numerical Optimization*. Springer Series in Operations Research. Springer.

ROCKAFELLAR, R. T. (1997). *Convex Analysis*. Princeton University Press.

SCHEEL, H. & SCHOLTES, S. (2000). *Mathematical programs with equilibrium constraints: stationarity, optimality, and sensitivity*. *Mathematics of Operations Research*, **25** 1–22.

SCHLENKRICH, S., GRIEWANK, A. & WALTHER, A. (2009a). *Global Convergence of quasi-Newton methods based on adjoint Broyden updates*. *Applied Numerical Mathematics*, **59** (5) 1120–1136.

SCHLENKRICH, S., GRIEWANK, A. & WALTHER, A. (2009b). *Local convergence analysis of TR1 updates for solving nonlinear equations*. Tech. Rep., Matheon Preprint 337.
URL `http://www.matheon.de/preprints/3864_tr1.pdf`

SCHLENKRICH, S., GRIEWANK, A. & WALTHER, A. (2010). *Local convergence analysis of adjoint Broyden methods*. *Mathematical Programming*, **121** 221–247. `http://www.springerlink.com/content/61727743m1w170u0/`.

SCHLOSSHAUER, V. (2009). *Strukturausnutzung und Speicherplatzbegrenzung für nichtlineare Optimierung*. Diplomarbeit, Humboldt-Universität zu Berlin.

SLAUGHTER, W. S. (2002). *The Linearized Theory of Elasticity*. Birkhauser.

SOKOLOWSKI, J. & ZOCHOWSKI, A. (1999). *On the Topological Derivative in Shape Optimization*. *SIAM J. Control Optim.*, **37** (4) 1251–1272.

SOKOLOWSKI, J. & ZOCHOWSKI, A. (2002). *Topological derivatives of shape functionals for elasticity systems*. *Int. Ser. Numer. Math.*, **139** 231–244.

STAUNING, O. (1997). *Introduction to FADBAD, a C++ Program package for Automatic Differentiation*. Talk in the course C0202 – Topics in Numerical Analysis. Technical University of Denmark.

TIMOSHENKO, S. (1928). *Vibration Problems in Engineering*. D. Van Nostrand Company.

VANDERBEI, R. J. (1995). *Symmetric quasidefinite matrices*. *SIAM Journal on Optimization*, **5** (1) 100–113.

WALTER, W. (2000). *Gewöhnliche Differentialgleichungen: Eine Einführung*, chap. 19–20. Springer, 7th edn.

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Berlin, den 23. Februar 2010 Kshitij Kulshreshtha